ТЕМА 3. Подання знань у формі клауз

Поняття клаузи Хорна. Співвідношення між клаузальною формою та стандартною формою логіки. Процедурна інтерпретація клауз Хорна. Основи пошуку розв'язків на мові клауз Хорна.

ПРАКТИЧНЕ ЗАНЯТТЯ №2. ПОДАННЯ ЗНАНЬ У ФОРМІ КЛАУЗ

Мета: оволодіти клаузальними засобами представлення знання про предметну область.

План заняття:

1. Формули логіки першого порядку.
2. Алгоритм приведення довільної формули числення предикатів до множини диз'юнктів.
3. Подання знань у формі клауз Хорна.

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 3.

ПРИКЛАДИ РОЗВ'ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. Знайти префіксну нормальну форму для формули

$$\forall x[P(x)\&\forall y\exists x(\neg Q(x,y)\rightarrow\forall zR(a,x,y))]$$

Розв'язання.

$$\forall x[P(x)\&\forall y\exists x(\neg Q(x,y)\rightarrow\forall zR(a,x,y))]\equiv$$
$$\forall x[P(x)\&\forall y\exists x((Q(x,y)\vee R(a,x,y))]\equiv$$
$$\forall x[P(x)\&\forall y\exists t(Q(t,y)\vee R(a,t,y)]\equiv$$
$$\forall x[P(x)\&\forall q\exists t(Q(t,q)\vee R(a,t,q))]\equiv$$
$$\mathbf{\forall x\forall q\exists t[P(x)\&(Q(t,q)\vee R(a,t,q))]}$$

Приклад 2. **.** Знайти сколемівську форму для формули

$$\exists u\forall v\exists w\forall x\forall y\exists zM(u,v,w,x,y,z)$$

Розв'язання. Їй відповідає сколемівська форма

$$\mathbf{\forall v\forall x\forall yM(a, v, f(v), x, y, g(v,x,y))}$$

де w замінена на $f(v)$ і z – на $g(v,x,y)$ – сколемівські функції.

Приклад 3. Знайти сколемівську форму і сколемівські функції для предикатної формули $\forall x\forall y\exists z\exists w\forall t(\neg S(x,y,y)\rightarrow(S(z,v,x)\&P(w,t,t)))$, де $S(x,y,z) = (x+y=z)$, $P(x,y,z) = (x*y=z)$ – предикати суми і добутку відповідно.

Розв'язання.

1) перетворення імплікації:

$$\forall x\forall y\exists z\exists w\forall t(S(x,y,y)\vee(S(z,v,x)\&P(w,t,t)))$$

2) виконуємо сколемівські перетворення, нехай $z = f(x,y)$, $w = g(x,y)$

$$S_A\colon\ \forall x\forall y\forall t(S(x,y,y)\vee(S(f(x,y),g(x,y),x)\&P(g(x,y),t,t)))$$

3) знаходимо сколемівські функції: f(x,y) + g(x,y) = x и g(x,y)∗t = t, тобто g(x,y) = 1 і f(x,y) = 1 – t.

Приклад 4. Перетворимо формулу $\forall x(P(x) \rightarrow \exists y(P(y) \vee \neg Q(x,y)))$ у еквівалентну множину диз'юнктів.

Розв'язання. Використовуємо алгоритм приведення довільної формули числення предикатів до множини диз'юнктів.

Перший крок. Приведемо початкову формулу до попередній нормальній форми. Елімінируємо **імплікацію** і отримаємо формулу $\forall x(\neg P(x) \vee \exists y(P(y) \vee \neg Q(x,y)))$. Винесемо змінну **y** за дужки: $\forall x \exists y(\neg P(x) \vee (P(y) \vee \neg Q(x,y)))$. Це можна зробити, тому що формула $\neg P(x)$ не залежить від змінної **y**. Якби вона залежала, то потрібно було б перейменувати зв'язану змінну **y**.

Другий крок. Проведемо сколемізацию отриманої формули. Лівіше за квантор існування є квантор загальності, значить, потрібно замінити всі входження змінної **y** новим унарним функціональним символом, залежним від **x**. Отримаємо формулу, що знаходиться в сколемівській нормальній формі: $\forall x(\neg P(x) \vee (P(f(x) \vee \neg Q(x,f(x))))$.

Третій крок. Елімінируємо квантор загальності: $\neg P(x) \vee (P(f(x)) \vee \neg Q(x,f(x)))$.

У четвертому і п'ятому кроках алгоритму необхідності немає, оскільки формула вже є диз'юнкт: $\neg P(x) \vee P(f(x)) \vee \neg Q(x,f(x))$.

Приклад 5. Сформувати знання у формі клауз Хорна щодо родини, дерево родинних зв'язків якої представлено на рис. 1.

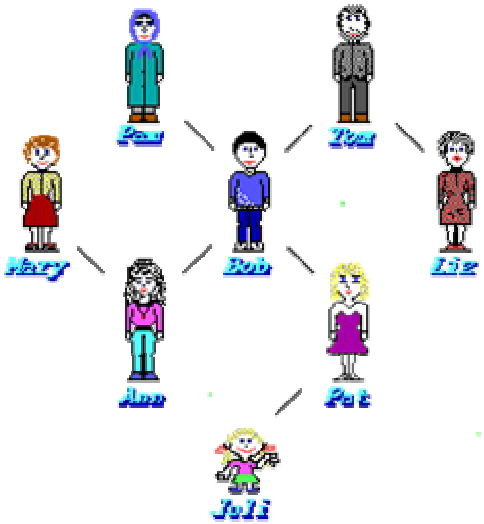
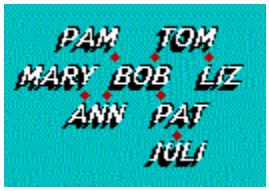


Рис. 1

Розв'язання. Введемо відношення "батьки" **(parent)** між об'єктами. Наприклад, **parent (tom, bob)** - це факт, що визначає, що **Том** є батьком **Боба**; **parent** - це ім'я відношення, **tom, bob** - це його аргументи. Тепер можна записати клаузи Хорна, що описують все дерево родинних зв'язків:

parent (pam, bob).
parent (tom, bob).
parent (tom, liz).
parent (bob, ann).
parent (bob, pat).
parent (mary, ann).
parent (pat, juli).



Введемо відношення "дитина" **child**, зворотне до **parent "**батьки". Можна було б визначити аналогічно: **child (liz, tom).** Але можна використати то, що відношення **child** зворотне до **parent,** і записати його у вигляді клаузи Хорна - **правила**:

child(Y, X):-parent (X, Y).

Правило читається так:

Для всіх X і Y
Y -child X, якщо
X -parent Y.

Правило відрізняється від факту тим, що факт завжди істинний, а правило описує твердження, яке буде істинним, якщо виконані певні умови.

Додамо ще одне унарне відношення, що визначає стать людини:

male(tom).
male(bob).
male(jim).
female(liz).
female(pam).
female(pat).
female(ann).

Тепер визначимо відношення **mother**. Воно описується таким чином:

Для всіх X Y
X -mother Y, if
X- parent Y і
X -female.

Таким чином, правило буде:

mother(X, Y):-parent(X, Y), female(X).

Визначимо відношення **sister:**

Для будь-яких X и Y
X sister Y, if
у X і Y є спільний батько,
і X female

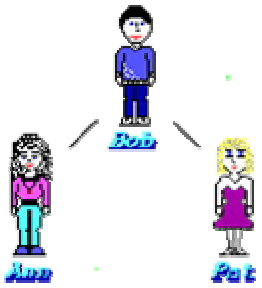
Запишемо правило у вигляді клаузи Хорна:

sister (X, Y):- parent(Z,X), parent(Z,Y), female(X).

Клауза Хорна (питання)

?-sister(ann, pat).

дає відповідь yes



ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

1. Знайти префіксну нормальну форму для формул:

1. $P(x) \vee (\forall x Q(x,z) \& \forall y R(y,z))$;
2. $\forall x(Q(x,z) \to R(x,t)) \vee \forall y \exists t(P(y,t) \sim S(y))$;
3. $\forall x(P(x,z) \Leftrightarrow Q(x,z)) \to \exists x \exists z(R(x,z) \to S(x,z))$.

2. Перетворимо наступні формули у еквівалентну множину диз'юнктів.

1. $(\neg \exists y P(y) \vee \forall x R(x)) \to \forall x(P(x) \to R(x))$
2. $\exists x(P(x) \to \forall y(P(y) \vee \neg Q(x,y)))$
3. $\forall x(P(x) \to \neg Q(x)) \wedge \forall x(P(x) \to R(x)) \to \exists x(R(x) \wedge \neg Q(x))$;
4. $\forall x(P(x) \to Q(x)) \wedge \forall x(Q(x) \to R(x)) \to \forall x(P(x) \wedge R(x))$
5. $\forall x(P(x) \to \neg Q(x)) \wedge \forall x(R(x) \to Q(x)) \to \exists x(\neg P(x) \wedge R(x))$
6. $(\forall x P(x) \to \exists x Q(x)) \sim \exists x(P(x) \to Q(x))$
7. $(\exists x P(x) \to \forall x Q(x)) \to \forall x(P(x) \to Q(x))$
8. $\exists x(P(x) \wedge Q(x)) \wedge \forall x(Q(x) \to \neg R(x)) \to \exists x(P(x) \wedge \neg R(x))$
9. $\forall x(P(x) \to Q(x)) \wedge \forall x(P(x) \to R(x)) \to \exists x(Q(x) \wedge R(x))$
10. $\forall x(P(x) \to \neg Q(x)) \wedge \forall x(R(x) \to P(x)) \to \exists x \neg(Q(x) \wedge R(x))$

3. Використовуючи предикати:

B(x) – «предмет **x** має чорний колір»;
W(x) – «предмет **x** має білий колір»;
C(x) – «предмет **x** – куб»;
S(x) – «предмет **x** - куля»;
D(x,y) – «предмет **x** розташований **під** предметом **y**»;
U(x,y) – «предмет **x** розташований **над** предметом **y**»;
L(x,y) – «предмет **x** розташований **зліва від** предмету **y**»;
R(x,y) – «предмет **x** розташований **праворуч від** предмету **y**»,

записати наступну думку:

«**Який би не був чорний куб, лежачий під всіма чорними кулями, зліва від нього не знаходиться жодної білої кулі»**

у формі клауз Хорна.

3. Записати знання у формі клауз Хорна щодо:

1. своєї родини;
2. своєї групи;
3. структури кафедри ІСПР;
4. структури факультету оподаткування;
5. дисциплін, які ви вивчаєте;
6. державного устрою країни;
7. розкладу навчальних занять;
8. відношень геометричних фігур;
9. податкового законодавства країни;
10. будови комп’ютера.

ТЕМА 4. Знання як об'єкти комп'ютерної обробки. Метод резолюції

Негативні цілі та твердження. Поняття підстановки та уніфікації виразів. Загальне правило резолюції. Резолюція зв'язків у графі сполучень. Глобальні стратегії пошуку розв'язків.

ПРАКТИЧНЕ ЗАНЯТТЯ №3. ЗНАННЯ ЯК ОБ'ЄКТИ КОМП'ЮТЕРНОЇ ОБРОБКИ. МЕТОД РЕЗОЛЮЦІЇ.

Мета: навчитися застосовувати метод резолюції для розв'язування найпростіших логічних задач.

План заняття

1. Застосування методу резолюції до знаходження логічних наслідків.
2. Знаходження розв'язків логічних задач методом резолюції.
3. Аналіз ефективності методів пошуку розв'язків при різних формах подання знань.

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 4.

Правило резолюції для логіки висловлень Графічно це правило можна зобразити так:

$$(A \vee P, B \vee \neg P)/A \vee B$$

Тут $A \vee P$ и $B \vee \neg P$ - батьківські диз'юнкти, P і $\neg P$ - контрарні літерали, $A \vee B$ - резольвента.

Якщо батьківські диз'юнкти складалися тільки з контрарних літералів, то резольвентою буде порожній диз'юнкт $\varnothing$.

ПРИКЛАДИ РОЗВ'ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. Задана множина диз'юнктів S:

1. $P \vee \neg Q_1 \vee \neg Q_2$
2. $P \vee \neg Q_3$
3. $U \vee \neg Q_3$
4. $U \vee \neg P \vee \neg R$
5. R
6. Q_1
7. Q_3

Довести методом резолюції, що диз'юнкт U є логічним наслідком з S.

Розв'язання. До множини S додається диз'юнкт

8. $\neg U$

Тоді з (3) і (8) має місце резольвента:

9. $\neg Q_3$

а з (7) і (9) – резольвента:

10. $\varnothing$ - порожній диз'юнкт,

що і потрібно було довести.

Приклад 2. Методом резолюції перевірити множину S диз'юнктів на суперечність:

$$S=\{p \vee q,\ p \vee r,\ \neg q \vee \neg r,\ \neg p\}.$$

Розв'язання. Перенумеруємо диз'юнкти:

1. $p \vee q$
2. $p \vee r$
3. $\neg q \vee \neg r$
4. $\neg p$

Обчислюємо і додаємо резольвенти (в дужках вказані № диз'юнктів)

5. q (1, 4)
6. r (2, 4)
7. $\neg q$ (3, 6)
8. $\varnothing$ (5, 7). Множина S диз'юнктів є суперечливою.

Приклад 3. Методом резолюції з'ясувати, чи є логічно правильним наступне просте міркування. Студент піде додому (p) або залишиться в університеті (q). Він не залишився в університеті. Отже, студент пішов додому. (Буквами позначені наявні в цьому міркуванні прості висловлення).

Розв'язання. Запишемо це міркування символічно за допомогою вказаних в дужках букв: $p \vee q$, $\neg q$, p. Істинність наслідку визначатиметься істинністю наявних висловів, $\{p \vee q, \neg q\} \models p$.

Застосуємо принцип дедукції: $\{p \vee q, \neg q, \neg p\} \models \varnothing$.

Суперечність множини доведемо за допомогою резолюції:

1. $p \vee q$,
2. $\neg q$,
3. $\neg p$,
4. p (1, 2).
5. $\varnothing$.(порожній диз'юнкт)

Можна вважати, що дане міркування є логічно правильним.

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

1. Чи є несуперечною множина висловлень:
   1) $\{\neg(A \to B),\ B \to A\}$;
   2) $\{A \wedge \neg B,\ \neg B \to \neg A\}$;
   3) $\{A \to B,\ A \to \neg B\}$;
   4) $\{A \to \neg A,\ \neg A \to A\}$;
   5) $\{A \to B,\ \neg A,\ \neg B\}$;

6) $\{A \to B, C \to B, A \wedge \neg C\}$;
7) $\{A \to \neg B, A \wedge B\}$;
8) $\{\neg A \to \neg B, C \to B, C \wedge \neg A\}$;
9) $\{A \leftrightarrow \neg B, \neg A \to \neg C, A \vee C, C \to B\}$.

2. Виразити умови завдання через фрази Хорна і провести докази, використовуючи метод резолюції.

1). Або Петро і Іван брати, або вони однокурсники. Якщо Петро і Іван брати, то Сергій і Іван не брати. Якщо Петро і Іван однокурсники, то Іван і Михайло також однокурсники. Отже, або Сергій і Іван не брати, або Іван і Михайло однокурсники.

2). Якщо Петро не зустрічав Івана, то або Іван не був на лекціях, або Петро бреше. Якщо Іван був на лекціях, то Петро зустрічав Івана, і Сергій був в читальному залі після лекцій. Якщо Сергій був в читальному залі після лекцій, то або Іван не був на лекціях, або Петро бреше. Отже, Іван не був на лекціях.

3). Наша футбольна команда або виграє матч, або програє, або зводить його до нічиєї. Якщо матч виграний або програний, то він не перенесений. Команда матч не виграла і не звела його до нічиєї. Отже, матч не перенесений і програний.

4). Якщо Джон не зустрічав цієї ночі Сміта, то або Джон був вбивцею, або Джон бреше. Якщо Сміт не був вбивцею, то Джон не зустрічав Сміта цієї ночі, і вбивство мало місце після півночі. Якщо ж вбивство мало місце після півночі, то або Сміт був вбивцею, або Джон бреше. Отже, Сміт був вбивцею.

5). Відомо, що хронічні сепульки завжди латентні або біфуркальні. Які з наступних тверджень в цьому випадку істинні:

a) сепульки не хронічні тільки у разі відсутності у них властивості латентності;

b) латентність сепулек не є необхідною умовою їх хронічності або біфуркальності;

c) хронічність сепулек є достатньою умовою їх латентності або біфуркальності;

d) для нехронічності сепулек необхідна відсутність у них як біфуркальності, так і латентності.

3. Перевірити методом резолюції, чи є несуперечною множина висловлень.

1) Ліда складе на «добре» екзамен з алгебри тоді і тільки тоді, коли вона не пропустить останньої лекції.

2) Ліда або пропустить останню лекцію, або не поїде на екскурсію до Києва.

3) Якщо Ліда не складе на «добре» екзамен з алгебри, вона не поїде на екскурсію до Києва.

4) Ліда складе на «добре» екзамен з алгебри або поїде на екскурсію до Києва.

4. Перевірити методом резолюції, чи є несуперечною множина висловлень.

1) Якщо Петро не поїхав у відрядження, то Таня не встигне написати листа додому.

2) Федір приїде тільки тоді, коли Таня встигне написати. листа додому.

3) Петро привезе нові книги, якщо він поїде у відрядження.

4) Якщо приїде Федір, то Петро не привезе нових книг.

5) Таня не встигла написати листа додому і Петро не привезе нових книг.

5. Слідчий допитує трьох обвинувачених *X, Y, Z. X* каже слідчому, щоб він не вірив свідченням *Y, Y* наполягає на тому, щоб слідчий не вірив свідченням *Z,* а *Z* стверджує, що ані *X,* ані *Y* не говорять правди. Хто з трьох обвинувачених говорить правду?

6. Грають у таку гру. В одній з двох шухляд сховано певну річ, і коло них є два учасники гри, які знають, де саме схована річ, причому один з них говорить тільки правду, другий — тільки неправду. Про це знає третій учасник гри, якому пропонується визначити, в якій саме шухляді знаходиться схована річ, використавши лише одне питання, на яке він може одержати тільки відповідь «так» чи «ні». Яке саме запитання він має сформулювати, щоб виконати поставлене завдання?

7. Леся запросила до себе подруг: Ганну, Віру і Олю. Відомо:

1) якщо прийде Ганна, то буде і Віра;

2) Ганна не буде тільки тоді, коли будуть Віра і Оля разом;

3) якщо не буде Олі, то не прийде і Ганна.

Хто з Лесиних подруг прийде до неї на запрошення?

8. Троє обвинувачених *X, Y, Z* дають такі свідчення *X* — *«Y* винен, а *Z* — ні»; *Y* — «Якщо *X* винен, то і *Z* теж»; а *Z*—«Я не винен, але хоч один з двох інших — винен».

1). Вважаючи, що *Y* і *Z* говорять правду, встановити, хто саме винен.

2). Якщо всі троє невинні, то хто з них сказав правду, а хто — неправду?

9. Задано 5 тверджень; треба визначити, яке з них є істинним, а яке хибним. Відомо, що:

1) серед цих 5 тверджень істинних є більше ніж хибних;

2) в списку цих 5 тверджень підряд слідують не більш ніж два твердження, які потребують однакової відповіді;

3) відповіді на перше і на п'яте питання — протилежні.

Якою має бути відповідь на друге питання, щоб правильні відповіді на всі поставлені питання визначались однозначно?

ТЕМА 5. Вступ до логічного програмування. Дескриптивний, процедурний і машинний зміст програми на мові Пролог

Зв’язок між логікою й програмуванням. Принципова відмінність Прологу від традиційних мов програмування. Програмування на Пролозі як процес створення системи фактів і правил, що характеризують розв'язуване завдання.

ПРАКТИЧНЕ ЗАНЯТТЯ №4. ДЕСКРИПТИВНИЙ, ПРОЦЕДУРНИЙ І МАШИННИЙ ЗМІСТ ПРОГРАМИ НА МОВІ ПРОЛОГ.

Мета: засвоїти основні логічні принципи програмування та виконання програм в системі Пролог.

План заняття:

1. Декларативна семантика Пролог-програми.
2. Процедурна семантика Пролог-програми.
3. Процедура обчислення цілей.

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 5.

Розрізняють **декларативну** і **процедурну семантику** (сенс, розуміння) Пролог-програми. Створюючи Пролог-програми, завжди треба пам'ятати про її процедурний і декларативний сенс.

Декларативний сенс стосується тільки відносин, визначених в програмі. **Декларативна семантика** визначає, що має бути результатом роботи програми, не вдаючись до подробиць, як це досягається.

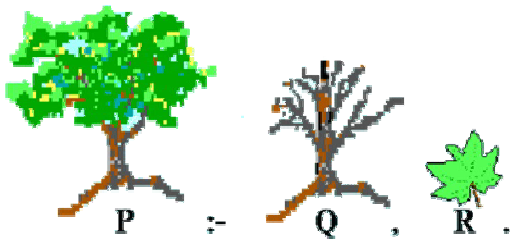
З іншого боку, **процедурний** сенс визначає, як цей результат може бути досягнутий, тобто як реально пропозиції обробляються Прологом.

ПРИКЛАДИ РОЗВ’ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Нехай задана пропозиція

P:-Q, R.

де **P, Q, R -терми.**



З погляду декларативного сенсу ця пропозиція читається так:

"P - істинно, якщо Q, R істинні" або **"Із Q і R випливає P."**

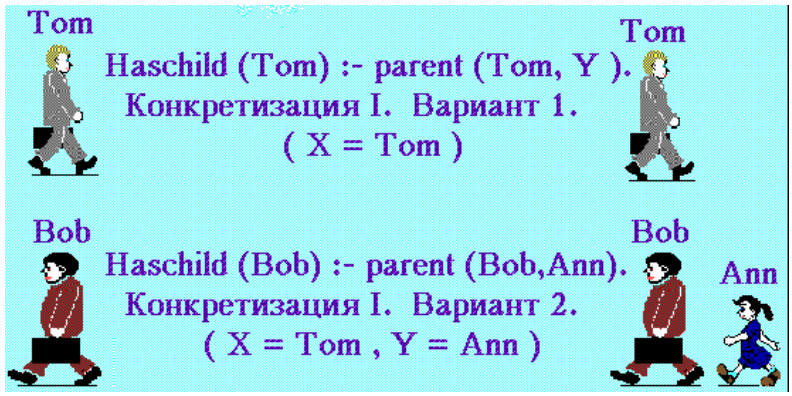
Тобто визначаються логічні зв'язки між головою пропозиції і цілями в його тілі.

Таким чином, **декларативний сенс програми** визначає, чи є дана мета істинною (досяжною), і якщо - так, то при яких значеннях змінних вона досягається.

Конкретизацією I пропозиції C називається результат підстановки в нього на місце кожної змінної деякого терма (відмітимо, що це відрізняється від конкретизації змінної).

Приклад 1 (конкретизація пропозиції)

haschild(X):-parent(X ,Y).
Пропозиція C.



Означення.

Хай дана деяка програма і мета **G.** Тоді, відповідно до декларативної семантики, можна стверджувати, що:

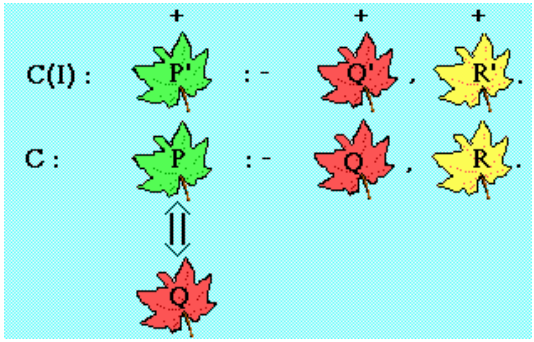
мета **G** істинна (тобто досяжна або логічно випливає з програми) тоді і тільки тоді, коли:

(1) У програмі існує пропозиція **C**, така, що:

(2) існує така його (**C**) конкретизація **I**, що:

(a) голова **I** збігається з **G** і

(b) всі цілі в тілі **I** істинні.



Наприклад:

female(ann).
parent(ann, bob).

C(I): mother(ann):-parent(ann, Y), female(ann).
C: mother(X) :-parent(X, Y), female(X).

?- mother(ann).

Це означення можна розповсюдити на питання таким чином.

У загальному випадку питання - список цілей, розділених комами.

Список цілей називається істинним (досяжним), якщо всі цілі в цьому списку істинні, досяжні, при однакових конкретизаціях змінних.

Кома між цілями означає кон'юнкцію цілей, і вони мають бути всі істинно. Можлива диз'юнкція цілей: істинна має бути принаймні одна з цілей. Дизъюнкция обозначается точкой с запятой ";".

Например: P :- Q; R. Читається: P істина, якщо Q - істина або R - істинна. Тобто, це те ж саме, що і клаузи Хорна:

P :- R.

P :- Q.

Кома зв'язує цілі сильніше, ніж крапка з комою. Таким чином, пропозиція:

P :- Q, R; S, T, U.

розуміється як

P :- (Q, R);(S, T, U).

і має такий сенс:

P :- Q, R.

P :- S, T, U.

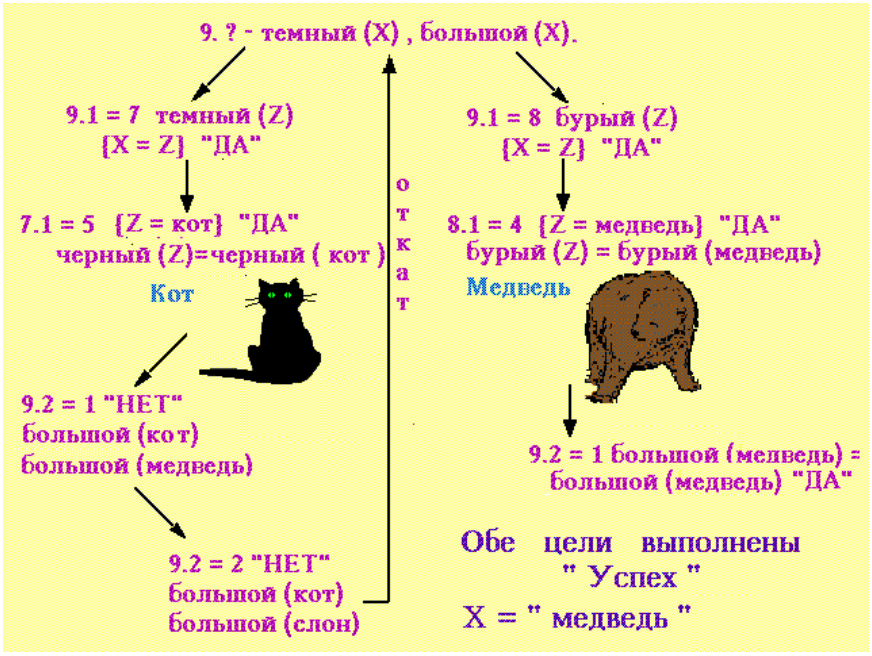
Процедурна семантика (процедурний сенс) Пролог-програми визначає, як Пролог-програма відповідає на питання. Відповісти на питання - це означає задовольнити цілі. Тому процедурна семантика Прологу - це процедура обчислення списку цілей з урахуванням програми.

Приклад 2 (процедурна семантика, трасування Пролог-програми)

Розглянемо Пролог-програму і на її прикладі - процедуру обчислення списку цілей.

1.большой(медведь).
2.большой(слон).
3.маленький (кот).
4.бурый(медведь).
5.черный(кот).
6.серый(слон).
7.темный(Z):-черный(Z).
7.1
8.темный(Z):-бурый(Z).
8.1
9.?-темный(X),большой(X).
9.1 9.2

Пропозиції перенумеровані для зручності.



Таким чином для обчислення цілей було потрібно 7 зіставлень і один відкіт.

Формальний опис процедури обчислення цілей

Нехай є список цілей:

G1	G2	. . .	Gm

1. Якщо список цілей порожній, то обчислення дає успіх, якщо ні, то виконаються пункт 2.

2. Береться перша мета **G1** із списку. Пролог вибирає з бази знань, переглядаючи її спочатку, першу пропозицію **C**,

C: H :- B1, B2, ..., Bn.

голова якої зіставляється з метою **G1**. Якщо такої пропозиції немає, то невдача. Якщо є, то змінні конкретизуються і мета **G1** замінюється на список цілей

B1'	B2'	...	Bn'

з конкретизованими значеннями змінних.

3. Розглядається рекурсивно через п.2 новий список цілей:

B1'	B2'	...	Bn'	G2	...	Gm

Якщо **C** -факт, то новий список укорочується на одну мету (n=0).

Якщо обчислення нового списку закінчується успішно, то і початковий список цілей виконується успішно.

Якщо ні, то новий список цілей відкидається, знімається конкретизація змінних і відбувається повернення до перегляду програми, але починаючи з пропозиції, наступної за пропозицією **C.**

Описаний процес повернення називається **бектрекинг (backtracking)**.

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

Визначити декларативний зміст, процедурну семантику та виконати трасування наступних Пролог-програм.

1.

```
length([], 0).
length([X|L], N):-length(L, M), N is M+1.
? length([a, b, c], N).
```

2.

```
DOMAINS
студент = symbol
то_що_можна_їсти = symbol
якість = symbol
PREDICATES
nondeterm любить(студент, то_що_можна_їсти)
має_вкус(то_що_можна_їсти, якість)
nondeterm їстівне(то_що_можна_їсти)
CLAUSES
любить(Петя, X):- їстівне(X), має_вкус(X, good).
має_вкус(піца, good).
має_вкус(плов, bad).
їстівне(плов).
їстівне(піца).
GOAL
любить(Петя, Що).
```

3.

```
data(one).
data(two).
data(three).
cut_test_a(X):- data(X).
cut_test_a('last clouse').
? cut_test_a(X),nl,write(X).
```

4.

```
minimum(X, Y, X):- X<=Y, !.
minimum(X, Y, Y):- Y.
? minimum(2, 5, Y).
```

5.

```
class(X, plus):-X>0, !.
class(X, minus):-X<0, !.
class(X, zero):-X=0, !.
```

```
? class(4, Y).
```

6.

```
plus(X,Y,Z):-integer(X),integer(Y),integer(Z),
        S is X + Y, S=Z.
plus(X,Y,Z):-integer(X),integer(Y),var(Z),
        Z is X + Y.
plus(X,Y,Z):-var(X),integer(Y),integer(Z),
        X is Z - Y.
plus(X,Y,Z):-integer(X),var(Y),integer(Z),
        Y is Z - X.
? plus(1,2,Z).
```

7.

```
нод(a,a,a).
нод(a,b,c):- БОЛЬШЕ(a,b), ВЫЧИТАНИЕ(a,b,d), нод(b,d,c).
нод(a,b,c):- БОЛЬШЕ(b,a), ВЫЧИТАНИЕ(b,a,d), нод(a,d,c).
ВЫЧИТАНИЕ(X,Y,Z):- СЛОЖЕНИЕ(Z,Y,X).
?нод(256,96,x).
```

8.

```
fib(0,_,0):- !.
fib(1,0,1):-!.
fib(N,G,F):- СЛОЖЕНИЕ(I,1,N), fib(I,H,G), СЛОЖЕНИЕ(H,G,F).
fi(L,K):- fib(L,_,K),!.
?fi(5,q).
```

9.

```
монах(н):- перенос(н,1,2,3).
перенос(0,а,б,_).
перенос(н,а,б,в):-
        БОЛЬШЕ(н,0),
        перенос(#н-1#,а,в,б),
        ВЫВОД(а,"->",б),
        перенос(#н-1#,в,б,а).
?монах(7).
```

10.

```
мати(Оля, Володя).
батько(Сергій, Володя).
мати(Оля, Сашко).
батько(Коля, Сашко).
мати(Люда, Сергій).
батько(Володя, Сергій).
батьки(х,у) :- мати(х,у).
```

```
батьки(x,y) :- батько(x,y).
бабуся(x,y) :- мати(x,z), батьки(z,y).
дідусь(x,y) :- батько(x,z),батьки(z,y).
? бабуся(x, Сашко).
```

11.

```
sqrt(0,0):-!.
sqrt(1,1):-!.
sqrt(x,y):- sqrt1(x,y,1.).
sqrt1(x,y,a):- РАВНО(#(x/a-a)/2.#,z), abs(z,b),
                БОЛЬШЕ(#b/a#,0.000005), !, sqrt1(x,y,#a+z#).
sqrt1(x,a,a).
abs(x,y):- МЕНЬШЕ(x,0),!,РАВНО(y,#-1*x#).
abs(x,x).
?sqrt(17.35,q).
```

12.

```
пузырек(а,б):- перест(а,в),!, пузырек(в,б).
пузырек(а,а).
перест([x,y|z],[y,x|z]):- БОЛЬШЕ(x,y).
перест([x|y],[x|z]):- перест(y,z).
?пузырек([47,43,44,45,5,4,3,2,1],a).
```

13.

```
вставсорт([],[]).
вставсорт([a|b],z):- вставсорт(b,c), встав(a,c,z).
встав(x,[a|b],[a|c]):- БОЛЬШЕ(x,a), !, встав(x,b,c).
встав(x,z,[x|z]).
?вставсорт([57,71,72,73,74,79,78,9,8,7,6,5,4,3,2,1],a).
```

14.

```
упорядочить([],[]).
упорядочить(a,[x|b]):- минэл(x,a), удалить(x,a,c), упорядочить(c,b).
минэл(x,[x]).
минэл(x,[x|y]):- минэл(z,y), НЕ(БОЛЬШЕ(x,z)),!.
минэл(z,[x|y]):- минэл(z,y).
удалить(x,[x|y],y):-!.
удалить(x,[a|y],[a|z]):- удалить(x,y,z).
?упорядочить([9,8,7,6,5,4,3,2,1],a).
```

ТЕМА 6. Побудова бази знань. Подання знань про предметну область у вигляді фактів й правил

Поняття бази знань у системі Пролог. Побудова бази знань як процес виявлення множини досліджуваних об'єктів і зв'язків між ними. Створення інформаційно-логічної моделі, що описується мовою Прологу. Методики та приклади розробки баз знань у вигляді множини фактів і правил.

ПРАКТИЧНЕ ЗАНЯТТЯ №5. ПОБУДОВА БАЗИ ЗНАНЬ.

Мета: засвоїти основні методи побудови баз знань та виконання програм в системі Пролог.

План заняття:

1. Методика створення інформаційно-логічних моделей.
2. Програмування та виконання програми в системі Пролог-Д.
3. Приклади побудови бази знань.

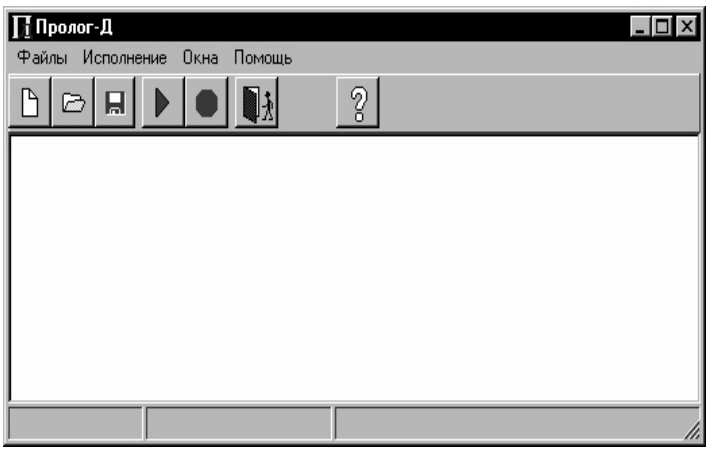
Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 6.

Для програмування та виконання програми будемо використовувати просту реалізацію концепції логічного програмування для навчального процесу - систему Пролог-Д.

Текст на Пролозі-Д містить повідомлення двох типів: факти й правила. Факт - це синтаксична конструкція, що дозволяє накопичувати інформацію, а правило - конструкція, за допомогою якої можна робити висновок або вивід. Об'єктами можуть бути елементами будь-якої природи: цілі числа, змінні, графічні образи тощо.

У такий спосіб факти й правила зв'язують об'єкти й відносини між ними і утворюють базу знань. На відміну від процедурних мов, порядок фактів і правил не має істотного значення для правильності результату, за винятком декількох особливих випадків.

Для запуску системи необхідно задати питання. Питання - це факт, якому передує символ "?". База знань і питання утворять програму мовою Пролог-Д.



Малюнок 1. Екран системи Пролог-Д в MS Windows.

Для запуску системи досить за допомогою миші вибрати ярлик Прологу-Д або за допомогою програм перегляду диска знайти файл із програмою prologd.exe і подвійним натисканням лівої клавіші миші ініціювати його виконання.

На екрані з'явиться заставка системи Пролог-Д Windows (малюнок 1).

Зображений на малюнку 1 інтерфейс є стандартним для систем, що працюють у середовищі Windows.

Управляти системою можна за допомогою маніпулятора миша, або за допомогою клавіатури. У першому випадку до потрібного місця на екрані підводить курсор миші й натискаючи ліву клавішу миші викликається необхідна дія. Якщо користувач вибрав один з елементів меню, то з'являється падаюче меню даного елемента основного меню.

Меню **Файли**

Система дозволяє працювати одночасно з декількома файлами. Число файлів обмежено тільки наявністю ресурсів Вашого комп'ютера. Кожному файлу відповідає одне вікно.

При виборі даного елемента меню на екрані з'являється наступне падаюче меню, зображене на малюнку 2.

Малюнок 2. Падаюче меню, що з'являється при виборі елемента Файли.

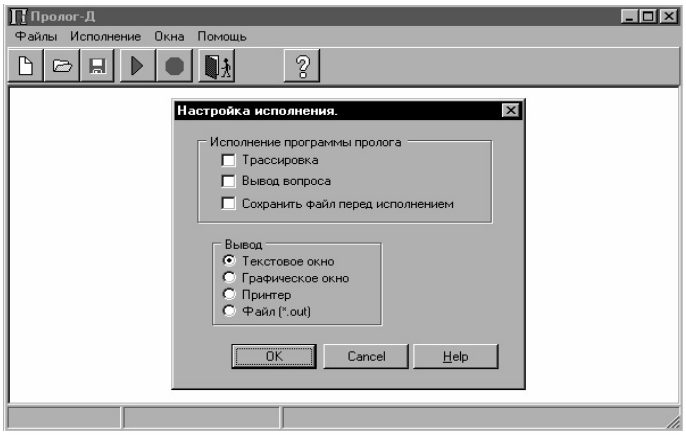
Елемент меню **Створити** дозволяє створювати новий файл, який буде оброблятися за допомогою системи Пролог-Д. При виборі даного елемента з'являється спеціальне вікно, що дозволяє створити потрібний файл.

Елемент меню **Відкрити** дозволяє відкрити або створити файл, що буде оброблятися за допомогою системи Пролог-Д. При виборі даного елемента з'являється спеціальне вікно, що дозволяє вибрати файл, що цікавить Вас, усередині даного каталогу.

Елемент меню **Зберегти** дозволяє записати оброблений файл на те ж місце. Для збереження обробленого файлу з будь-яким ім'ям у довільному місці пам'яті необхідно використати елемент меню **Зберегти як**. При виборі даного елемента меню з'являється вікно, з допомогу, якого можна вибрати ім'я файлу, диск і каталог на диску, у якому записується інформація.

Меню **Виконання**

Містить елементи **Виконати, Перервати** й **Настроювання.** На малюнку 3 показана панель настроювання.

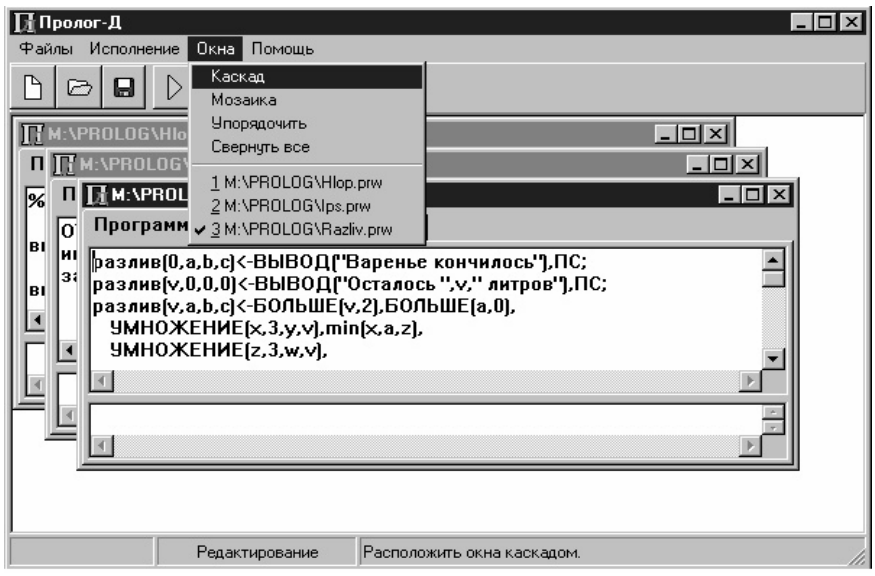


Малюнок 3. Панель настроювання режимів виконання Пролог-програми.

Відзначаючи або знімаючи позначки у відповідних квадратиках панелі настроювання можна включити й виключити трасування, вивід питання, доручити системі зберігати текст програми при кожному запуску, а також вибрати, куди направляти вивід у ході виконання програми.

Меню **Вікна**

При виборі цього елемента з'являється падаюче меню, зображене на малюнку 4.



Малюнок 4. Падаюче меню, що з'являється при виборі елемента Вікна.

Меню **Допомога**

При виборі даного елемента меню користувачеві надається можливість вибір різних видів допомоги, що вказують у падаючому меню: допомога про програму й допомога по розділах.

ПРИКЛАДИ РОЗВ'ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. Обчислити число $x = 2 \times 3 + 1$.
Розв'язання. Для цього досить набрати на клавіатурі питання:

?МНОЖЕННЯ(2,3,1,x); (<Enter>)

відповідь системи: x=7

Приклад 2. Побудувати базу знань "Батьки".
Розв'язання. Набрати за допомогою клавіатури текст такої програми:

мати(Людмила, Сашко);(<Enter>)
батько(Сергій, Сашко);(<Enter>)
бабуся(Надя, Сашко);(<Enter>)
онук(x, y)<-бабуся(y, x);(<Enter>)

До неї можна поставити запитання:

?мати(x, Сашко);(<Enter>),

що означає змістовно - "Як кличуть маму Сашка?".

Відповідь системи: x = Людмила

Натисніть на клавішу <Esc> і база знань знову з'явиться на екрані. Можна спробувати задати ще кілька питань, наприклад:

?бабуся(x, y);(<Enter>),
?онук(x, y);(<Enter>),
?батько(Сергій, x);(<Enter>).

Приклад 3. Побудувати базу знань "Родинні зв'язки".
Розв'язання. Наприклад, можна визначити наступні правила:

бабуся(x,y)<- мати(x,z),мати(z,y);
бабуся(x,y)<- мати(x,z),батько(z,y);
дідусь(x,y)<- батько(x,z),батько(z,y);
дідусь(x,y)<- батько(x,z),мати(z,y);

Приклад 4. Побудувати базу знань "Родина".
Розв'язання. До записаних вище правил додамо кілька фактів і правил:

мати(Оля, Володя);
батько(Сергій, Володя);
мати(Оля, Сашко);
батько(Коля, Сашко);
мати(Люда, Сергій);
батько(Володя, Сергій);
батьки(x,y)<- мати(x,y);
батьки(x,y)<- батько(x,y);

бабуся(x,y)<- мати(x,z),батьки(z,y);
дідусь(x,y)<- батько(x,z),батьки(z,y);

У даному прикладі для визначення поняття батьки(x,y) треба було більш одного правила. По суті справи, тут використане недетерміноване розгалуження, що дає альтернативне визначення цього відношення й яке використовується системою після того, як було застосовано перше правило. Варто підкреслити, що у визначенні беруть участь обидва правила. У загальному випадку число правил не обмежено.

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

1. Побудуйте базу знань своєї родини та реалізуйте її в системі Пролог-Д.
2. Побудуйте базу знань, що описує країни Європи, та реалізуйте її в системі Пролог-Д.
3. Побудуйте базу знань щодо структури кафедри ІСПР та реалізуйте її в системі Пролог-Д.
4. Побудуйте базу знань щодо структури факультету оподаткування та реалізуйте її в системі Пролог-Д.
5. Побудуйте базу знань дисциплін, які ви вивчаєте, та реалізуйте її в системі Пролог-Д.
6. Побудуйте базу знань про державний устрій країни, та реалізуйте її в системі Пролог-Д.
7. Побудуйте базу знань про розклад навчальних занять та реалізуйте її в системі Пролог-Д.
8. Побудуйте базу знань про відношення геометричних фігур та реалізуйте її в системі Пролог-Д.
9. Побудуйте базу знань щодо податкового законодавства країни та реалізуйте її в системі Пролог-Д.
10. Створіть базу знань щодо будови комп’ютера та реалізуйте її в системі Пролог-Д.
11. Побудуйте базу знань для фактів вигляду УЧЕНЬ(прізвище, предмет, оцінка), запропонувавши правила для предикатів:
УСПІШНИЙ(прізвище), НЕУСПІШНИЙ(прізвище), ВІДМІННИК(прізвище), ГУМАНІТАРІЙ(прізвище), ФІЗМАТ(прізвище) та реалізуйте її в системі Пролог-Д.
12. Побудуйте базу знань для фактів вигляду МІСТО(назва, населення), запропонувавши правила МІСТЕЧКО(назва), СЕРЕДНЄ(назва), МЕГАПОЛІС(назва) та реалізуйте її в системі Пролог-Д.

ТЕМА 7. Арифметика й інші убудовані предикати мови Пролог

Убудований арифметичний предикат МНОЖЕННЯ. Убудований предикат "відсікання" для керування логічним виводом. Убудовані предикати графіки та інші убудовані предикати.

ПРАКТИЧНЕ ЗАНЯТТЯ №6. АРИФМЕТИКА Й ІНШІ УБУДОВАНІ ПРЕДИКАТИ МОВИ ПРОЛОГ.

Мета: засвоїти основні прийоми розв’язування математичних задач та використання вбудованих предикатів мови Пролог-Д.

План заняття

1. Програмування математичних задач в системі Пролог-Д.
2. Використання предиката "відсікання".
3. Приклади використання інших убудованих предикатів.

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 7.

Системи логічного програмування, до числа яких відноситься й Пролог-Д, не призначені для обчислень. Тому традиційні дії, пов'язані з виконанням арифметичних операцій, здійснюються за допомогою спеціальних убудованих предикатів. У системі Пролог-Д для виконання арифметичних дій передбачений один убудований арифметичний предикат: МНОЖЕННЯ(Арг1,Арг2,Арг3,Арг4). Убудований предикат МНОЖЕННЯ має чотири аргументи, допускає обертання всіх аргументів, однак, він може бути використаний тільки як ціль в реченні. Предикат МНОЖЕННЯ передбачає реалізацію формули: Арг1*Арг2+Арг3=Арг4.

ПРИКЛАДИ РОЗВ’ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. За допомогою предиката МНОЖЕННЯ описати інші арифметичні дії.

Розв’язання. Предикат МНОЖЕННЯ передбачає обертання аргументів і реалізує арифметичні операції в області цілих чисел, передбачених синтаксисом вхідної мови (від -32767 до 32767). Наступна база знань мовою Прологу-Д описує інші арифметичні дії.

```
ДОДАВАННЯ(X,Y,Z)<-МНОЖЕННЯ(1,X,Y,Z);

ВІДНІМАННЯ(X,Y,Z)<-МНОЖЕННЯ(1,X,Z,Y);

МНОЖЕННЯ(X,Y,Z)<-МНОЖЕННЯ(X,Y,0,Z);

ДІЛЕННЯ(X,Y,Z)<-МНОЖЕННЯ(Y,Z,0,X);
```

У всіх чотирьох випадках X,Y - суть операнди операцій, а Z - результат. Наприклад, ДОДАВАННЯ(X,Y,Z) реалізує арифметичну операцію додавання: Z=X+Y. Предикат МНОЖЕННЯ дозволяє описувати обчислювальні завдання.

Приклад 2. Скласти програму на Пролозі-Д для обчислення площі прямокутника, що має сторони довжиною a й b. Відома формула площі прямокутника $S_{пр} = ab$.

Розв'язання. Предикат повинен мати три аргументи - довжини сторін і величину площі. Ім'я предиката повинне відбивати його призначення; ймовірно цьому критерію задовольнить ім'я площа:

МНОЖЕННЯ(X,Y,Z)<-МНОЖЕННЯ(X,Y,0,Z);
площа(a,b,S)<-МНОЖЕННЯ(a,b,S);

Перший предикат МНОЖЕННЯ треба було визначити для наочності запису. Відмітимо, що предикат площа обертається, тобто користуючись цим описом можна обчислити не тільки площу по заданих сторонах, але й кожну (одну) сторону по іншій стороні й площі. До бази знань можна поставити питання:

?площа(10,20,S);

відповідь системи Пролог-Д: S=200,

?площа(a,20,100);

відповідь системи Пролог-Д: a = 5.

Приклад 3. Скласти програму на Пролозі-Д для обчислення об'єму паралелепіпеда висотою h, у основі якого прямокутник, що має сторони довжиною a й b. Відома формула об'єму паралелепіпеда $V_{пр} = abh$.

Розв'язання. Предикат, що буде виконаний, якщо буде обчислений об'єм паралелепіпеда, повинен мати чотири аргументи - довжини сторін a, b, висоту h і величину об'єму. Ім'я предиката повинне відбивати його призначення, імовірно, цьому критерію задовольнить ім'я "об'єм".

МНОЖЕННЯ(X,Y,Z)<-МНОЖЕННЯ(X,Y,0,Z);
об'єм(a,b,h,V)<-МНОЖЕННЯ(a,b,S), МНОЖЕННЯ(S, h, V);

Як і раніше, предикат об'єм є оборотний, тобто використовуючи цей опис можна обчислити не тільки об'єм по заданих сторонах і висоті, але й кожну (одну) сторону або висоту по висоті, стороні й об'єму.

До бази знань можна задати питання: ?об'єм(10,20,5,V); відповідь системи Пролог-Д: V=200.

Приклад 4. Опишіть мовою Пролог обчислення функції Хевисайда, задану формулою:

$$h(x) = \begin{cases} 0, \text{ якщо } x \le 0 \\ 1, \text{ якщо } x > 0 \end{cases}.$$

Розв'язання. База знань повинна містити опис предиката МЕНШЕ Й ДОРІВНЮЄ та предиката, що виконується при обчисленні функції Хевисайда (буде називатися ХЕВИСАЙД).

Предикат ХЕВИСАЙД буде мати два аргументи, перший - це аргумент функції, а другий - її значення, та буде визначаться через два альтернативних описи для всіх значень х.

МИР(X,Y):- НЕ(БІЛЬШЕ(X,Y));
ХЕВИСАЙД(X,0):- МИР(X,0);
ХЕВИСАЙД(X,1):- БІЛЬШЕ(X,0);

Задамо, наприклад, питання:

?ХЕВИСАЙД(10,X);

відповідь системи Пролог: X=1.

Приклад 5. Скласти програму на Пролозі-Д для побудови кута з вершиною в точці (x,y).

Розв'язання.

кут(x,y)<-ЛІНІЯ(x,y,10,10,1), ЛІНІЯ(x,y,150,50,1);
?кут(100,100);.

Спочатку буде намальований відрізок, що з'єднує точки (100,100) і (10,10), а потім відрізок, що з'єднує точки (100,100) і (50,50). Якби п'ятим аргументом предикатів ЛІНІЯ було б число рівне нулю, то точки відрізків були б не видимі.

Не обов'язково, щоб опис всієї картинки був записаний в одному реченні. Частина опису може бути виділена у вигляді окремого речення. Наприклад, попередню програму можна модифікувати так:

кут(x,y)<-ЛІНІЯ(x,y,10,10,1),продовження(x,y);
продовження(x,y)<-ЛІНІЯ(x,y,50,50,1);
?кут(100,100);.

Нова програма буде виконувати ті ж самі функції, хоча й записується у два речення.

Приклад 6. Скласти програму на Пролозі-Д для побудови вектора, що виходить із точки А з координатами (x, y) у точку В координатами (s,t).

Розв'язання.

вектор($A(x,y), B(s,t)$)<-ЛІНІЯ(x,y,s,t,1);

Таким чином, система Пролог-Д допускає можливість використання змінних у графічних примітивах. Необхідно відзначити особливість графічних об'єктів, що описуються за допомогою змінних. У процесі роботи системи може виявитися, що якась змінна в описі графічного примітива не визначена. У цьому випадку графічний примітив однаково буде виконаний, однак змінна приймає всі припустимі для неї значення. Іншими словами на екрані з'явиться геометричне місце точок, що задається рівнянням графічного об'єкта.

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

1. Скласти програму на Пролозі-Д обчислення площ геометричних фігур: трапеції, трикутника, паралелограма.

2. Скласти програму на Пролозі-Д обчислення площі й довжини кола. Яка точність обчислень цих величин? Чи можна обчислити радіус кола за його довжиною?

3. Скласти програму на Пролозі-Д для обчислення функції, що задана співвідношенням:

$$F(x)=\begin{cases} x, \text{ якщо } x < -1, \\ x+1, \text{ якщо } -1 < x < 1, \\ x, \text{ якщо } x > 1. \end{cases}$$

4. Скласти програму на Пролозі-Д, що будує прямокутний трикутник.

5. Скласти програму на Пролозі-Д, що підраховує величину податку за величиною доходу.

6. Скласти програму на Пролозі-Д, що встановлює зв’язок по закону Ома (зв'язок між струмом, напругою і опором) для 2 резисторів - послідовних і паралельних.

7. Скласти програму на Пролозі-Д, що малює олімпійські кільця.

8. Скласти програму на Пролозі-Д, що малює автостоянку з різнокольоровими машинами.

9. Скласти програму на Пролозі-Д, що відображає пісочні часи з піском, що перетікає.

10. Скласти програму на Пролозі-Д, що відображає секундомір с бігучою стрілкою.

ТЕМА 8. Рекурсія та структури даних у програмах на мові Пролог

Застосування рекурсії для опису завдань при роботі із системами логічного програмування. Приклади створення бази знань для рекурсивних програм. Типові структури даних у програмах на мові Пролог.

ПРАКТИЧНЕ ЗАНЯТТЯ №7. РЕКУРСІЯ ТА СТРУКТУРИ ДАНИХ У ПРОГРАМАХ НА МОВІ ПРОЛОГ.

Мета: засвоїти основні принципи побудови та виконання рекурсивних програм.

План заняття

1. Рекурсивні означення.
2. Запис рекурсії на мові Пролог.
3. Виконання рекурсивної програми на мові Пролог.
4. Вплив порядку речень у базі знань на виконання програми.

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 8.

Існує величезна кількість завдань, у яких відносини між об'єктами можна визначити тільки використовуючи самі обумовлені співвідношення. При цьому маємо правила, що називають рекурсивними. Застосування рекурсії для опису завдань при роботі із системами логічного програмування є широко розповсюдженим прийомом.

ПРИКЛАДИ РОЗВ'ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. Скласти програму на мові Пролог-Д для обчислення найбільшого загального дільника (НЗД) двох чисел.

Розв'язання. Предикат, що виконується, якщо знайдено НЗД двох даних чисел буде мати ім'я НЗД і три аргументи: числа a, b і значення НЗД - c. Для опису обчислення НЗД використаються наступні міркування.

По-перше, якщо a = b, то c = a = b. По-друге, якщо a > b, то необхідно обчислити НЗД для чисел b й a – b. По-третє, якщо b > a, то необхідно обчислити НЗД для чисел a й b - a. Предикат НЗД є оберненим.

Ці три твердження природно можуть бути записані на Пролозі-Д:

```
НЗД(a,a,a);
НЗД(a,b,c) :- БІЛЬШЕ(a,b), ВІДНІМАННЯ(a,b,d), НЗД(b,d,c);
НЗД(a,b,c) :- БІЛЬШЕ(b,a), ВІДНІМАННЯ(b,a,d), НЗД(a,d,c);
ВІДНІМАННЯ(X,Y,Z) :- МНОЖЕННЯ(1,X,Z,Y);
```

Якщо до цієї бази знань поставити запитання:

```
?НЗД(24,18,x);
```

відповідь системи Пролог:

```
x=6 ІНШИХ РОЗВ'ЯЗКІВ НЕМАЄ
```

Приклад 2. Скласти програму на мові Пролог-Д для обчислення послідовності Фибоначчи $f_0 = 0$, $f_1 = 1$, $f_n = f_{n-1} + f_{n-2}$ при $n > 1$.

Розв'язання. Перша формула $f_0 = 0$ відповідає твердженню про те, що значення нульового елемента послідовності дорівнює нулю. Це можна записати у вигляді факту: Фиб(0,0);. Друга формула $f_1 = 1$ відповідає твердженню: перший елемент дорівнює 1. На Пролозі це можна записати так: Фиб(1,1);. Третя формула $f_n = f_{n-1} + f_{n-2}$ є запис рекурсивного співвідношення:

```
Фиб(N,X) :- БІЛЬШЕ(N,1), ВІДНІМАННЯ(N,1,M),
            ВІДНІМАННЯ(N,2,ДО), Фиб(M,Y), Фиб(K,Z),
            ДОДАВАННЯ(Y,Z,X);
```

ВІДНІМАННЯ й ДОДАВАННЯ - імена предикатів віднімання й додавання, заданих за допомогою правил:

```
ДОДАВАННЯ(X,Y,Z):- МНОЖЕННЯ(1,X,Y,Z);
ВІДНІМАННЯ(X,Y,Z):- МНОЖЕННЯ(1,X,Z,Y);.
```

Дані речення складають базу знань мовою Пролог, що дозволяє обчислювати значення елементів послідовності. У відповідь на запитання:

```
?Фиб(6,X);
```

відповідь системи Пролог:

```
x=8 ІНШИХ РОЗВ'ЯЗКІВ НЕМАЄ
```

Приклад 3. Це завдання засновано на класичній головоломці про ханойського або буддійського монаха, що переставляє диски до кінця світа. Виконавець працює з трьома стержнями і декількома дисками різного розміру, що надягають на стержень. Він може знімати верхній диск з будь-якого стержня, щоб перенести його на інший стержень, при цьому не можна класти більший диск на менший. Мета – переставити всі диски (згідно легенді - 64) з одного на іншій (використовуючи один допоміжний) – і тоді обов'язково що-небудь відбудеться – чи то кінець світу, чи то навпаки.

Припустимо, що стержні мають номери 1, 2, 3. Тоді наш виконавець розуміє команди типу "XY", що означає – узяти диск із X і перенести його на Y. Отже, для перестановки 2 дисків із стержня 1 на стержень 2 він повинен виконати наступні команди: 13, 12, 32.

Скласти на Пролозі-Д рекурсивну програму, що описує послідовність дій при перенесенні N дисків із 1 на 2 з використанням стрижня 3.

Розв'язання.

```
монах(н):- перенос(н,1,2,3).
перенос(0,а,б,_).
перенос(н,а,б,в):-
        БІЛЬШЕ(н,0),
        перенос(#н-1#,а,в,б),
        ВИВІД(а,"->",б),
        перенос(#н-1#,в,б,а).
?монах(7).
```

Використання рекурсивних означень дає можливість записати базу знань більш лаконічно. Рекурсія може бути використана й для створення графічного об'єкта, що динамічно змінюється.

Для цього на тому самому місці послідовно фіксується образ об'єкта, так, що його кольори поперемінно змінюються від кольору фона до кольору, обумовленого в базі знань.

Наприклад, опис птаха, що летить та махає крилами. У предикаті змах описаний змах униз й, потім, змах нагору. Першому положенню відповідає горизонтальне положення відрізка білих кольорів, потім цей відрізок гаситься, стає кольором фона. Положенню нагору відповідає кут, що складається із двох відрізків білих кольорів, які потім гасяться.

Кожне з понять нагору й униз описується окремо, при цьому виконуються арифметичні дії, необхідні для обчислення координат початку й кінця відрізків.

Періодичне повторення змаху нагору й униз здійснюється за допомогою рекурсивного звертання до тому самому предиката птах.

Приклад 4. Скласти програму на мові Пролог-Д, що зображує політ птаха.

Розв'язання. Повний опис птаха, що махає крильми наведено нижче:

```
птах(x,y)<-змах(x,y),птах(x,y);
змах(x,y)<-униз(x,y,1),униз(x,y,0), нагору(x,y,1), нагору(x,y,0);
униз(x,y,c)<-сдв(x,y,z,t,u,v), ЛІНІЯ(z,y,u,y,1);
нагору(x,y)<-сдв(x,y,z,t,u,v), ЛІНІЯ(x,y,z,t,1), ЛІНІЯ(x,y,u,v,1);
сдв(x,y,z,t,u,v)<-сдп(x,y,z,t),сдл(x,y,u,v);
сдп(x,y,z,t)<-ДОДАВАННЯ(x,5,z),ДОДАВАННЯ(t,5,y);
сдл(x,y,z,t)<-ДОДАВАННЯ(z,5,x),ДОДАВАННЯ(t,5,y);
```

Декларативність мови дозволяє створювати досить складні картини, використовуючи відомі принципи декомпозиції графічного об'єкта на частини й наступний їхній опис.

Приклад 5. Скласти програму на мові Пролог-Д, що малює будинок та пташку, яка летить.

Розв'язання. Наведена нижче програма дає приклад складного синтетичного об'єкта та використання рекурсії для визначення динамічної зміни об'єкта:

```
будинок(x,y)<-дах(x,y,r,f), поверх(x,y,f);
поверх(x,y,f)<-ДОДАВАННЯ(y,20,z), квадр(x,y,f,z);
дах(x,y,r,f)<-ДОДАВАННЯ(r,10,y), ДОДАВАННЯ(x,20,f), трик(x,y,r,f);
трик(x,y,z,t)<-ЛІНІЯ(x,y,t,y,1), РОЗПОДІЛ2(x,t,f), кут(x,y,z,t,f);
кут(x,y,z,t,f)<-ЛІНІЯ(x,y,f,z,1), ЛІНІЯ(f,z,t,y,1);
РОЗПОДІЛ2(x,t,f)<-ДОДАВАННЯ(x,t,r), РОЗПОДІЛ(r,2,f);
квадр(x,y,z,t)<-ЛІНІЯ(x,y,x,t,1), ЛІНІЯ(x,y,z,y,1), ч2(x,y,z,t);
ч2(x,y,z,t)<-ЛІНІЯ(x,t,z,t,1), ЛІНІЯ(z,y,z,t,1);
```

птах(x,y)<-змах(x,y), птах(x,y);
змах(x,y)<-униз(x,y,1),униз(x,y,0),нагору(x,y,1),нагору(x,y,0);
униз(x,y,c)<-сдв(x,y,z,t,u,v), ЛІНІЯ(z,y,u,y,1);
нагору(x,y)<-сдв(x,y,z,t,u,v), ЛІНІЯ(x,y,z,t,1), ЛІНІЯ(x,y,u,v,1);
сдв(x,y,z,t,u,v)<-сдп(x,y,z,t), сдл(x,y,u,v);
сдп(x,y,z,t)<-ДОДАВАННЯ(x,5,z), ДОДАВАННЯ(t,5,y);
сдл(x,y,z,t)<-ДОДАВАННЯ(z,5,x), ДОДАВАННЯ(t,5,y);

У відповідь на питання:

?будинок(70,110), птах(120,50);

на екрані буде побудовані будинок і намальована пташка, що махає крилами.

Даний приклад є сукупністю двох описів - будинку й птаха, що махає крилами. Зверніть увагу на одна обставину - предикат птах повинен бути описаний після предиката будинок, тому що в ньому є кінцева рекурсія - типу петлі.

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

1. Скласти програму на мові Пролог-Д для обчислення найменшого загального кратного двох чисел.

2. Скласти програму на мові Пролог-Д для скорочення раціональної дробі $\frac{p}{q}$.

3. Скласти програму на мові Пролог-Д для обчислення найбільшого загального дільника трьох чисел.

4. Скласти програму на мові Пролог-Д для обчислення найменшого загального кратного трьох чисел.

5. Скласти програму на мові Пролог-Д для обчислення суми $S = 1 + 2 + ... + n$.

6. Скласти програму на мові Пролог-Д для обчислення суми $S = 1^2 + 2^2 + ... + n^2$.

7. Скласти програму на мові Пролог-Д для обчислення послідовності $f_0 = 1$, $f_1 = 0$, $f_n = 2f_{n-1} - f_{n-2}$ при $n > 1$.

8. Скласти програму на мові Пролог-Д для обчислення послідовності $f_0 = -1$, $f_1 = 0$, $f_n = f_{n-1} + f_{n-2} - 1$ при $n > 1$.

9. Скласти програму на мові Пролог-Д для обчислення послідовності $f_0 = 0$, $f_1 = 2$, $f_n = f_{n-1} - 2f_{n-2}$ при $n > 1$.

10. Скласти програму на мові Пролог-Д для обчислення послідовності $f_0 = 0$, $f_n = f_{n-1} + 2n$ при $n > 0$.

11. Використовуючи рекурсивне означення, напишіть базу знань, що описує багатоповерховий будинок.

ТЕМА 9. Обробка списків на мові Пролог

Поняття та приклади списків. Типові операції над списками: приналежність елемента до списку, склеювання двох списків, сортування та обертання списку, знаходження та видалення елемента списку. Приклади баз знань на мові Пролог, що обробляють спискові структури.

ПРАКТИЧНЕ ЗАНЯТТЯ №8. ОБРОБКА СПИСКІВ НА МОВІ ПРОЛОГ.

Мета: навчитися складати програми в системі Пролог-Д для обробки списків різної структури.

План заняття

1. Рекурсивне означення списків.
2. Основні операції над списками.
3. Сортування списків.

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 9.

ПРИКЛАДИ РОЗВ'ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. Скласти програму на мові Пролог-Д для сортування списку повним перебором.

Розв'язання.

```
сортування("Список","УпорСпис"):-
        перестановка("Список","УпорСпис"),
        упорядкований("УпорСпис")!.

перестановка("Список"["Перший"|"Інші"]):-
        виділити("Перший","Список","НовийСписок"),
        перестановка("НовийСписок","Інші").
перестановка([],[]).

виділити("Елемент",["Елемент"|"Інші"], "Інші").

виділити("Елем"["ДрЕлем"|"Спис"]["ДрЕлем"|"ДрСпис"]):-
        виділити("Елем","Спис","ДрСпис").
впорядкований(["Елемент"]).

впорядкований(["Перший","Другий"|"Інші"]):-
        НЕ(БІЛЬШЕ("Перший","Другий")),
        впорядкований(["Другий"|"Інші"]).

?сортування([8,7,6,5,4,3,2,1], а).
```

Приклад 2. Скласти програму на мові Пролог-Д для сортування списку методом бульбашки.

Розв'язання.

```
бульбашка(а,б):- перест(а,в),!, бульбашка(в,б).
бульбашка(а,а).
```

```
перест([x,y|z],[y,x|z]):- БІЛЬШЕ(x,y).
перест([x|y],[x|z]):- перест(y,z).

? бульбашка([47,43,44,45,5,4,3,2,1],a).
```

Приклад 3. Скласти програму на мові Пролог-Д для сортування списку методом вставки.

Розв'язання.

```
вставсорт([],[]).
вставсорт([a|b],z):- вставсорт(b,c), встав(a,c,z).
встав(x,[a|b],[a|c]):- БІЛЬШЕ(x,a), !, встав(x,b,c).
встав(x,z,[x|z]).

?вставсорт([57,71,72,73,74,79,78,9,8,7,6,5,4,3,2,1],a).
```

Приклад 4. Скласти програму на мові Пролог-Д для сортування списку методом перестановки.

Розв'язання.

```
упорядкувати([],[]).
упорядкувати (a,[x|b]):- мінел(x,a), віддалити(x,a,c), упорядкувати(c,b).
мінел(x,[x]).
мінел(x,[x|y]):- мінел(z,y), НЕ(БІЛЬШЕ(x,z)),!.
мінел(z,[x|y]):- мінел(z,y).
віддалити(x,[x|y],y):-!.
віддалити(x,[a|y],[a|z]):- віддалити(x,y,z).

? упорядкувати([9,8,7,6,5,4,3,2,1],a).
```

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

1. Скласти програму на мові Пролог-Д обчислення суми елементів заданого числового списку.

2. Скласти програму на мові Пролог-Д обчислення середнього арифметичного елементів заданого числового списку.

3. Скласти програму на мові Пролог-Д, що породжує список, що містить натуральні числа від 1 до N (за збільшенням).

4. Скласти програму на мові Пролог-Д, що породжує список, що містить квадрати натуральні чисел від 1 до N (за збільшенням).

5. Скласти програму на мові Пролог-Д, що дозволяє видалити із заданого списку всі від'ємні числа.

6. Скласти програму на мові Пролог-Д, що замінює в заданому списку всі від'ємні числа нулями.

7. Скласти програму на мові Пролог-Д, що замінює в заданому списку всі числа їх квадратами.

8. Скласти програму на мові Пролог-Д, що з’ясовує, що даний елемент є N-м в заданому списку.

9. Скласти програму на мові Пролог-Д, що замінює в заданому списку два однакові елементи, що стоять поряд, одним.

10. Скласти програму на мові Пролог-Д, що встановлює, що два задані списки не мають спільних елементів.

11. Скласти програму на мові Пролог-Д, що встановлює, що один список є підсписком іншого, тобто всі елементи одного списку містяться в іншому списку.

12. Скласти програму на мові Пролог-Д, що дозволяє розділити заданий список на два підсписки, - у перший помістити всі позитивні числа з початкового списку, в другій - від'ємні, а нулі - відкинути.

13. Скласти програму на мові Пролог-Д, яка по двох заданих списках утворює третій, куди поміщує елементи, які присутні в обох списках.

14. Скласти програму на мові Пролог-Д, яка по двох заданих списках створює третій, куди поміщує ті елементи, які не є спільними для заданих списків.

15. Скласти програму на мові Пролог-Д, яка по двох заданих числових списках створює третій, такий, що складається з сум відповідних пар елементів з початкових списків.

16. Скласти програму на мові Пролог-Д, що визначає, що в заданому списку заданий елемент зустрічається більше одного разу.

17. Скласти програму на мові Пролог-Д, яка для заданого списку визначає скільки разів в ньому зустрічається заданий елемент.

ТЕМА 10. Подання знань за допомогою семантичних мереж

Способи задання семантичних мереж. Види семантичних відносин: ієрархічні, функціональні, кількісні, просторові, часові, атрибутні, логічні, лінгвістичні. Інтернет як розподілена база знань та семантична мережа глобального масштабу. Семантичні мережі у системах штучного інтелекту (системах машинного перекладу, експертних системах тощо).

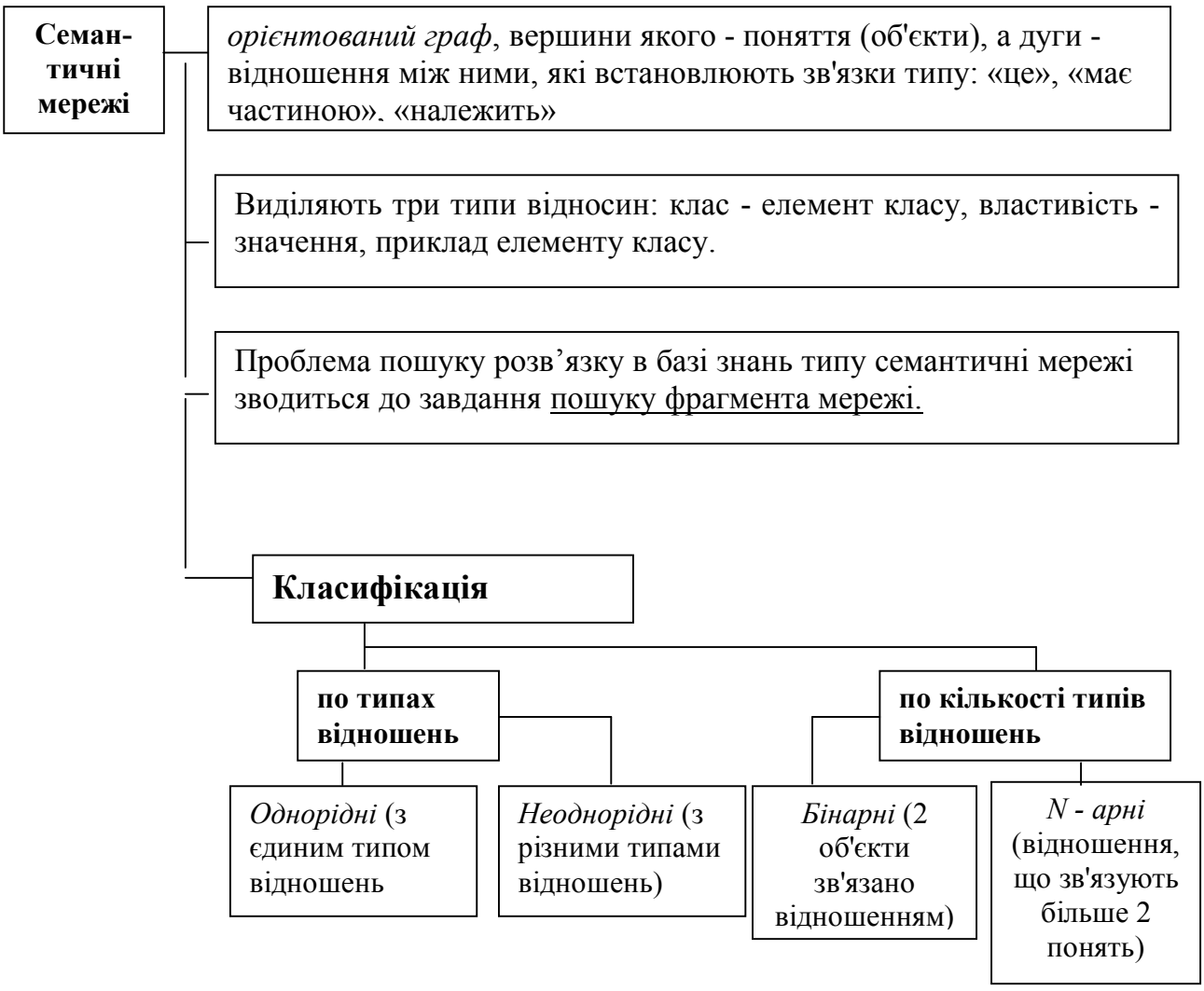
ПРАКТИЧНЕ ЗАНЯТТЯ №9. ПОДАННЯ ЗНАНЬ ЗА ДОПОМОГОЮ СЕМАНТИЧНИХ МЕРЕЖ.

Мета: засвоїти основні підходи до аналізу та побудова бази знань певної предметної області у вигляді семантичної мережі.

План заняття.

1. Способи задання семантичних мереж.
2. Представлення знань у вигляді семантичних мереж.
3. Приклади семантичних мереж.

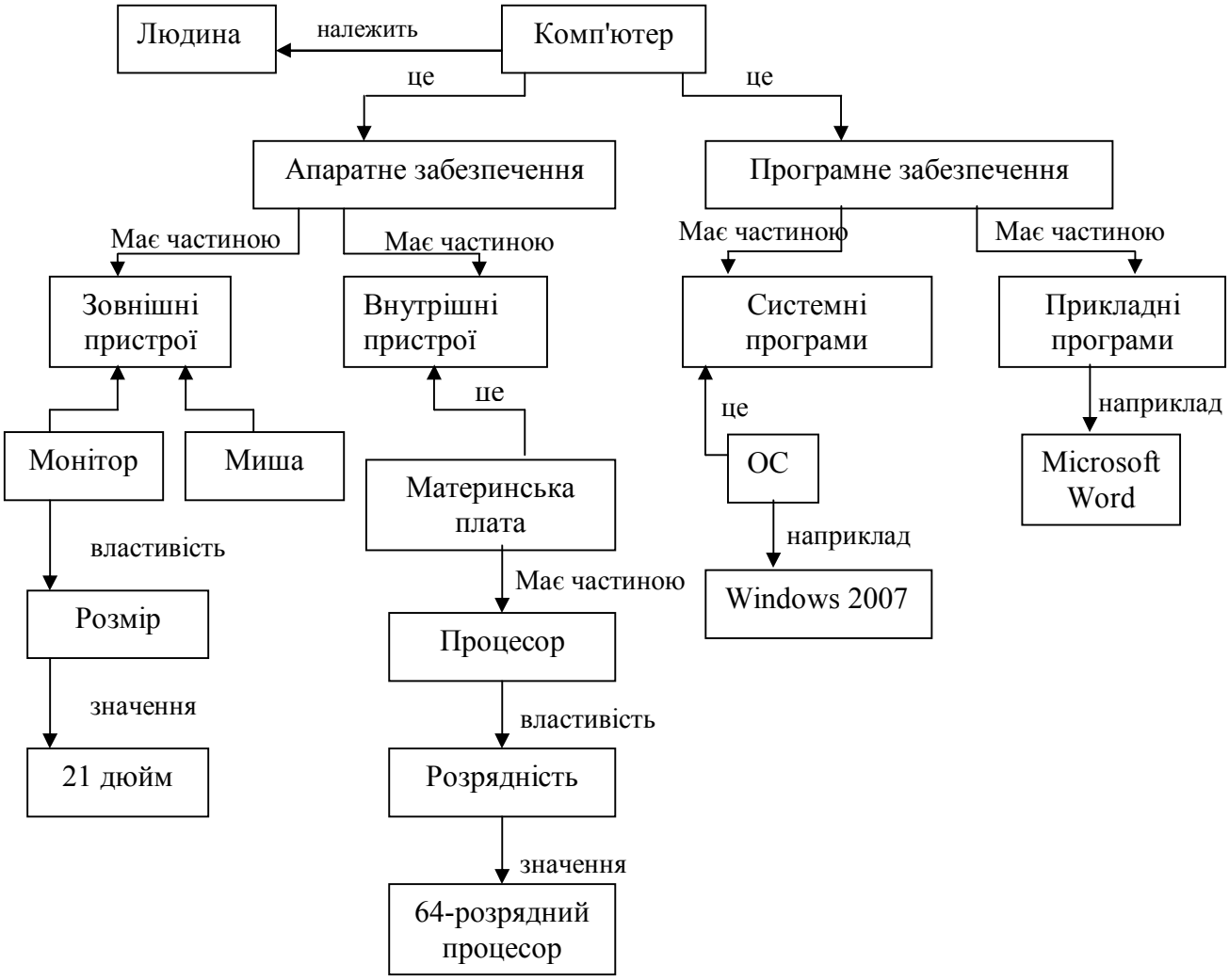
Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 10.



ПРИКЛАДИ РОЗВ’ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. Представити знання про комп’ютерну систему у вигляді семантичної мережі. Як вершини взяті поняття: Людина, Комп'ютер, Апаратне забезпечення, Програмне забезпечення, Зовнішні пристрої, Монітор, Миша, Внутрішні пристрої, Материнська плата, Процесор, Системні програми, Операційна система, Прикладні програми.

Розв’язання.



Мал. 1. Семантична мережа.

Використовуючи ознаки класифікації семантичних мереж, визначимо її видові характеристики. Дана семантична мережа є неоднорідною (з різними типами відношення: «це», «має частиною», «належить») і N-арною (відношення зв'язують більше двох понять).

Приклад 2. Представити тезаурус у вигляді семантичної мережі.

Розв’язання. У найпростішому випадку тезаурусом є словник синонімів: кожному входу (текстовому виразу, поняттю) зіставляється понятійний клас (концепт). У багаторівневому, або ієрархічному, тезаурусі окремі концепти групуються в крупніші класи. Поняття можуть бути

пов'язані між собою; їх взаємозв'язок відображають відношеннями, специфічними для даного тезауруса.

Структура семантичної мережі: вершини, що є науковими і діловими термінами, пов'язані бінарними відношеннями; терміни, приписані одній вершині, пов'язані відношенням синонімії.

Вершини зв'язані між собою стосунками типу "це-є" і "відноситься-к". Понятійними називаються вершини, які є родо-видовими або мають вхідне відношення асоціації хоч би для одного терміну.

Семантична мережа представлена у вигляді текстового файлу, в якому роздільником є кінець рядка, а кожен рядок є полем вершини.

Для визначення типу поля використовуються наступні мітки: ":П:", ":Ч:", ":О:", ":Т:", ":Е:". Розглянемо формат даних на прикладі декількох вершин:

:П:

:Ч: Біотоп

:О: Біосфера

:Т: Екосфера

:Е: Ecosphere

:П:

:Ч: Матеріальне виробництво

:Т: Серійне виробництво

:Е: Serial production

:П: Державна реєстрація нерухомості

:О: Нерухоме майно

:Т: Державна реєстрація нерухомості

Державна реєстрація прав на нерухоме майно

Державна реєстрація операцій з нерухомим майном

:Е: State registration of immovable property

State registration of the immovable property righs

State registration of the immovable property deals

Кожна вершина починається з мітки ":П:". У цьому рядку міститься найменування поняття для понятійної вершини.

Поле з міткою ":Ч:" визначає родо-видовую зв'язок (зв'язок "це-є"). У приведеному вище прикладі поняття "Серійне виробництво" є підкласом поняття "Матеріальне виробництво".

Поле з міткою ":О:" визначає асоціативний зв'язок (зв'язок "відноситься-к"). Наприклад, поняття "Державна реєстрація нерухомості" асоціативно відноситься до поняття "Нерухоме майно".

Поле з міткою ":T:" містить термін російською мовою, а поле з міткою ":E:" – на англійському. Терміни представлені як окремими словами, так і словосполуками.

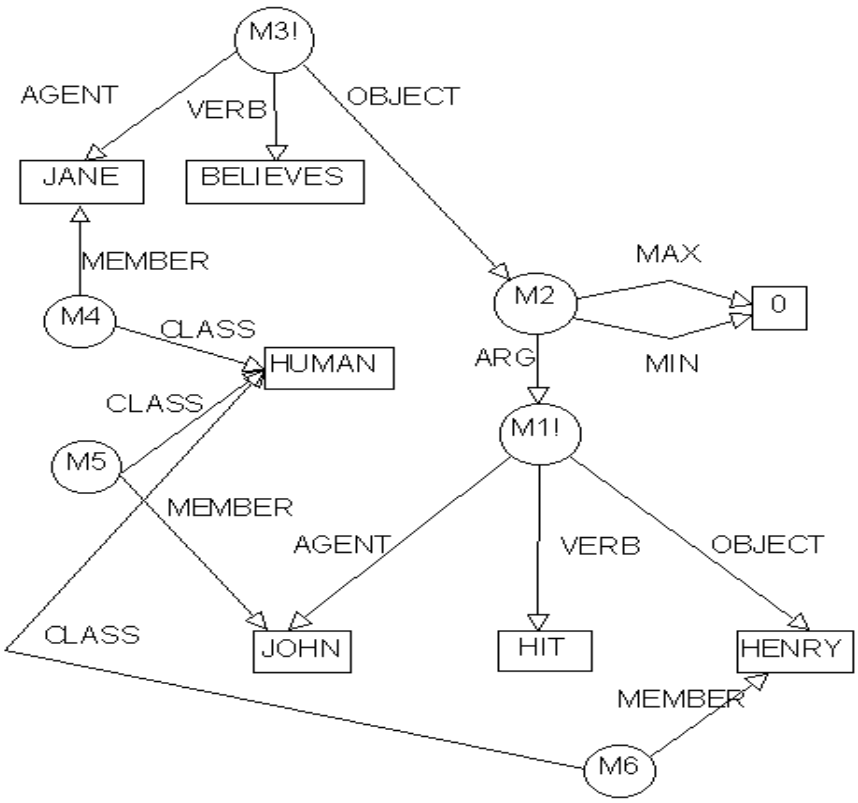
Кожна вершина може містити декілька полів одного типу, що починаються з нового рядка без мітки.

Побудова простого тезауруса здійснена оголошенням всіх вершин класами, що складаються зі всіх термінів даної вершини. Другим етапом було оголошення класами всіх понятійних вершин із занесенням в підкласи даного концепту всіх пов'язаних з ним вхідним родо-видовим або асоціативним відношенням термінів.

Таким чином, побудований тезаурус класичного вигляду з ієрархічною структурою і двома представленнями (з впорядковуванням на основі класифікації і за абеткою).

Приклад 3. Представити фразу '*Jane believes that John didn't hit Henry, but he did.*' у вигляді семантичної мережі.

Розв'язання.



Мал. 2. Семантична мережа фрази 'Jane believes that John didn't hit Henry, but he did.'

Малюнок 2 містить представлення фрази '*Jane believes that John didn't hit Henry, but he did.*'. У прямокутниках позначені базові вузли; молекулярні вузли і зразки - в колах. Вузол M1! містить пропозицию '*John hit Henry,*'. Вузол M2, використовуючи оцінку упевненості, яку можна приписати пропозиції, містить заперечення справедливості M1!. Вузол M3! відповідає

пропозиції упевненості агента в пропозиції M2. Пропозиції M4, M5, M6 містять твердження про приналежність вузлів, що позначають Jane, John і Henry, класу Human.

Все знання про окремі об'єкти, класи об'єктів, їх властивості, абстракції, дії, пропозиції, правила і мета-правила представлене у вигляді мережі, що складається з вузлів і направлених іменованих відношень між ними. Вузол в мережі належить одному з чотирьох видів: базовий, змінний, молекулярний і зразок - і має унікальний ідентифікатор. Базовий вузол відповідає елементу, концептуально відмінному від будь-якого іншого вузла мережі. Змінні вузли відповідають довільним елементам мережі і можуть бути ототожнені з деяким базовим. З базових і змінних вузлів не можуть виходити дуги, їх властивості визначені дугами, що входять в них. Молекулярні вузли відповідають пропозиціям (зокрема, і правилам), зразки (patterns) - функціональним термам з вільними змінними і використовуються для пошуку на мережі.

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

1. Зобразити семантичну мережу, де як вершини виступають поняття: Людина, Шевченко, Таврія, Автомобіль, Вид транспорту, Двигун. Використовуючи ознаки класифікації семантичних мереж, визначити її видові характеристики.

2. Зобразити семантичну мережу ситуації «Отримання студентом книги в бібліотеці», де як вершини виступають поняття: Книга, Студент, Бібліотека, Назва бібліотеки, Автор книги, Назва книги, Місце розташування бібліотеки. Використовуючи ознаки класифікації семантичних мереж, визначити її видові характеристики.

3. Зобразити семантичну мережу ситуації «Отримання студентом заліку», де як вершини виступають поняття: Викладач, Студент, Бібліотека, Кафедра, Залікова Книжка, Дисципліна, Рейтинг, Назва факультету, Питання, Аудиторія, Група. Використовуючи ознаки класифікації семантичних мереж, визначити її видові характеристики.

4. Зобразити семантичну мережу, де як вершини виступають поняття: Знання, Семантична мережа, Пропозиції, Фрейми, Пролог, Експертна система, Комп’ютер, Мова програмування, Предикат. Використовуючи ознаки класифікації семантичних мереж, визначити її видові характеристики.

5. Представити структуру НУДПСУ у вигляді семантичної мережі.

6. Представити структуру факультету оподаткування НУДПСУ у вигляді семантичної мережі.

7. Представити структуру кафедри ІСПР факультету оподаткування НУДПСУ у вигляді семантичної мережі.

8. Представити структуру ДПС України у вигляді семантичної мережі.

9. Представити структуру мови Пролог у вигляді семантичної мережі.

10. Представити структуру дисципліни МЗПЗ у вигляді семантичної мережі.

ТЕМА 11. Подання знань за допомогою фреймів

Особливість фрейм-підходу до проблеми подання знань. Поняття, структура та властивості фреймів (слоти, процедури та правила, наслідування, статичність та динамічність). Класифікація фреймів (фрейм-образ, фрейм-сценарій). Конкретизація, ієрархія та наслідування фреймів. Способи формалізування фреймів. Фрейми та об'єктно-орієнтоване програмування. Приклади формалізованого представлення фреймів-сценаріїв. Механізми «пристосування» фрейму до реальної ситуації. Використання фреймів в евристичному пошуку. Недоліки фреймових систем.

ПРАКТИЧНЕ ЗАНЯТТЯ №10. ПОДАННЯ ЗНАНЬ ЗА ДОПОМОГОЮ ФРЕЙМІВ.

Мета: засвоїти поняття, структуру, класифікацію та властивості фреймів, підходи до подання знань за допомогою фреймових моделей.

План заняття

1. Конкретизація змісту слотів.
2. Побудова ієрархій та властивостей наслідування фреймів.
3. Формування фреймових сценаріїв предметної області.

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 11.

Структура фрейму включає три основних типи даних: поняття (назва фрейму), характеристика (назва термінала — вершини нижнього рівня), значення характеристики (заповнювач термінала).

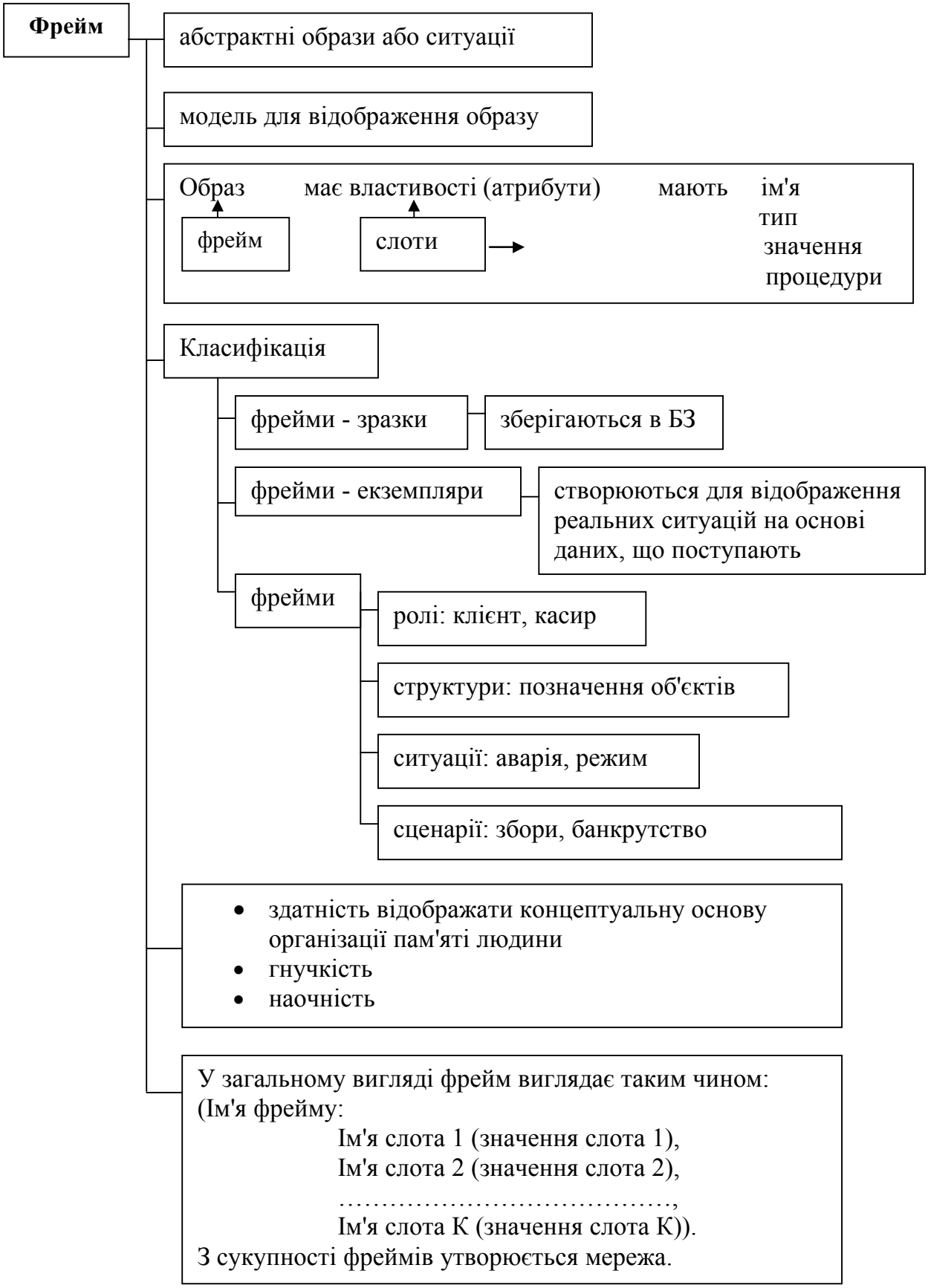
Фрейм надає засоби організації знань в слотах, що містять характеристики та структури. В моделі фрейму - це щось на зразок схеми з категоріями і підкатегоріями. Фрейм - це абстрактний образ для представлення деякого стереотипу сприйняття.

Слоти визначають атрибути або процедурні знання, пов'язані з його атрибутами, для поняття, представленого фреймом. Кожен слот може містити один або більше фасетів. Фасети описують тип значень, дозволені значення, число значень та інші властивості значень, яких може набувати слот. Фасети (або їх ще деколи називають підслотами) описують деякі знання або процедури атрибуту в слоті.

Більшість систем штучного інтелекту використовують набір фреймів, що з'єднані один з одним певним чином і творять певну ієрархію. Однією з найбільш важливих властивостей фреймів в таких ієрархіях є наслідування властивостей. Фрейм-потомок містить фактичні значення атрибутів-слотів, ті ж самі, як в батьківському фреймі, який подає більш загальний опис сутності.

ПРИКЛАДИ РОЗВ'ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. Побудувати структурно-логічну схему поняття "фрейм".
Розв'язання на мал. 1.



Мал. 1. Структурно-логічна схему поняття "фрейм".

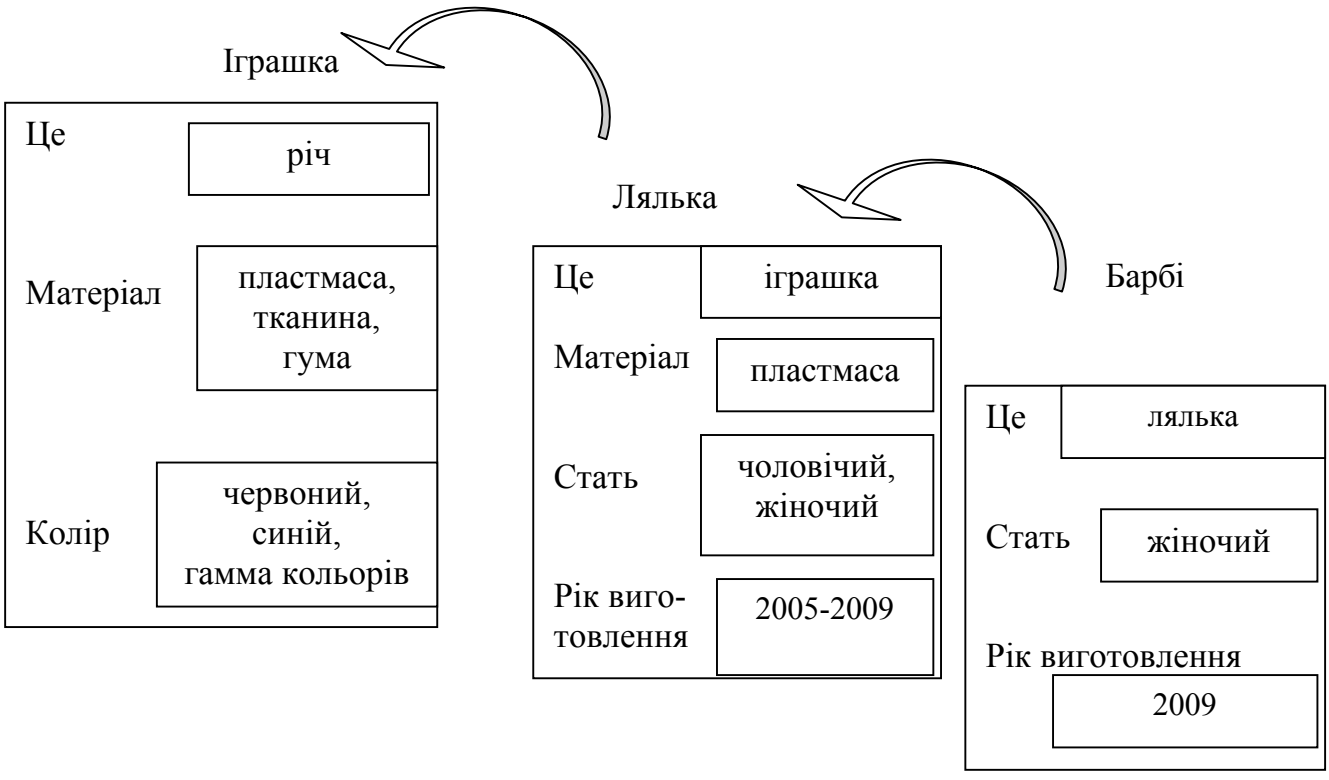
Приклад 2. Скласти фрейм з ім'ям Кондитерські вироби, де іменами слотів є Назва виробу, Термін зберігання, Виробник, Країна-виробник, Маса-нетто, а в дужках перераховані значення для п'яти слотів.

Розв'язання.

(Кондитерські вироби:
Назва виробу (Булка - Бісквіти - Пряник),
Термін зберігання (10 діб - 1 місяць - 15 діб),
Виробник (Хлібозавод №1 - Фірма «Порошенко» - Хлібозавод №3),
Країна-виробник (Україна - Україна - Польща),
Маса-нетто (30 грама - 300 грама - 90 грама))

Приклад 3. Побудувати мережу фреймів «Іграшка», «Лялька», «Барбі». Поняття «Барбі» успадковує властивості фреймів «Іграшка» і «Лялька», які знаходяться на більш високому рівні ієрархії. Спадкоємство властивостей може бути частковим, так, жіноча стать успадковується для «Барбі» з фрейму «Лялька».

Розв'язання.



Мал. 2. Мережа фреймів «Іграшка», «Лялька», «Барбі».

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

Побудувати мережі фреймів, що складається з понять:

1. Людина, Дитина, Учень.
2. Транспорт, Автомобіль, Запорожець.
3. Університет, факультет, група.
4. Установа, відділ, відділення.
5. Математика, алгебра, теорія матриць.
6. Комп’ютер, материнська плата, процесор.
7. Вулиця, дім, квартира.
8. Ліс, дерево, листя.
9. Держава, влада, Верховна Рада.
10. Речовина, рідина, вода

визначивши успадковані властивості для них.

Побудувати структурно-логічні схеми понять:

1. Фрейму-ролі.
2. Фрейму-сценарії.
3. Фрейму-ситуації.
4. Просторового фрейму.

Скласти фрейм з ім'ям Книга, де іменами слотів є Автор, Назва книги, Об’єм, Видавництво, Рік видання.

Скласти фрейм з ім'ям Курс, де іменами слотів є Номер курсу, Кількість студентських груп, Куратор, Кафедра, Старости груп.

Скласти фрейм з ім'ям Дисципліна, де іменами слотів є Назва дисципліни, Кількість годин, Література, Викладач, Підсумковий контроль.

Скласти фрейм з ім'ям Комп’ютер, де іменами слотів є Операційна система, Процесор, Оперативна пам’ять, Материнська плата, Зовнішня пам’ять.

Скласти фрейм з ім'ям Потяг, де іменами слотів є Номер потяга, Вагон, Купе, Вартість квітка, Час відправлення.

Скласти фрейм з ім'ям Студент, де іменами слотів є Ім’я студента, Адреса, Назва вузу, Назва спеціальності, Курс..

ТЕМА 12. Продукційні моделі подання знань

Поняття та властивості продукційної моделі. Загальна структура продукції та її компоненти (ядро, умова та сфера застосування тощо). Процедури управління системою продукції. Класифікація ядер продукції. Стратегії організації пошуку розв'язків. Пряме та зворотне виведення. Типові дисципліни виконання продукції. Основні стратегії вирішення конфліктів у продукційних системах.

ПРАКТИЧНЕ ЗАНЯТТЯ №11. ПРОДУКЦІЙНІ МОДЕЛІ ПОДАННЯ ЗНАНЬ.

Мета: засвоїти принципи побудови продукційної моделі предметної області, конкретизації структури продукції та її компонентів, формування процедур управління системою продукції та засоби комп'ютерної реалізації продуктивних моделей.

План заняття

1. Побудова продукційної моделі предметної області.
2. Конкретизація структури продукції.
3. Засоби комп'ютерної реалізації продуктивних моделей.

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 12.

Продукційною системою називатимемо будь-яку сукупність продукції. У загальній формі продукції мають вигляд

П, Р, А=>В, Q.

Тут А=>В - звичайна продукція «Якщо..., то...», яка носить назву *ядра продукції.* Елемент Р характеризує зовнішні умови або *умови застосовності* продукції, визначувані чинниками, що не входять безпосередньо в А, наприклад цілями, які стоять перед міркуючою системою.

Елемент П характеризує *сферу проблемної області* бази знань або *передумови застосовності* продукції. Нарешті, Q характеризує постумови продукції, вказуючи на ті зміни, які необхідно внести до бази знань і в систему продукції після реалізації даної продукції.

Всі частини, окрім ядра, є необов'язковими.

Продукційна модель представлення знань є розвитком логічних моделей у напрямі ефективності представлення і виведення знання. Найбільше застосування для реалізації продукційних моделей отримала мова ПРОЛОГ.

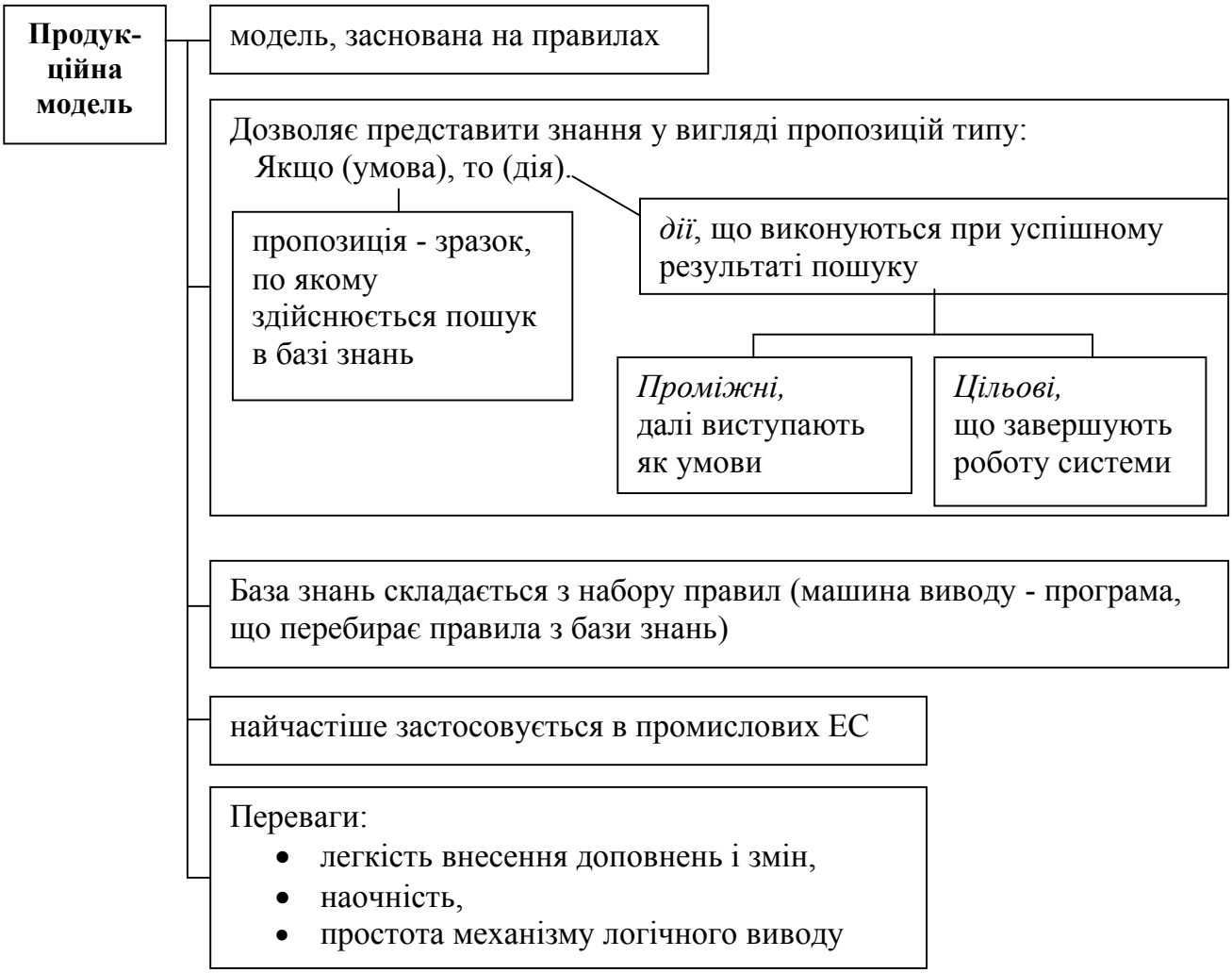
Продукційна модель найчастіше застосовується в промислових експертних системах. Вона привертає увагу розробників своєю наочністю, високою модульністю, легкістю внесення доповнень і змін і простотою механізму логічного виводу.

Недоліки продукційних моделей: малий ступінь структуризації бази знань, неясність взаємних відношень продукції, неуніверсальність.

ПРИКЛАДИ РОЗВ'ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. Побудувати структурно-логічну схему поняття "Продукційна модель".

Розв'язання на мал. 1.



Мал. 1. Структурно-логічна схема поняття "Продукційна модель".

Приклад 2. Припустимо, ми проводимо технічний огляд автомобіля і хочемо знати, поїде він чи ні. Ми перевіряємо акумулятор і стартер і на основі результатів огляду ухвалюємо рішення. Ось наші правила:

Правило 1. Якщо акумулятор сів або несправний стартер, то двигун не заведеться.

Правило 2. Якщо акумулятор заряджений, справний стартер, то двигун заведеться.

Правило 3. Якщо двигун заведеться, то автомобіль поїде.

Правило 4. Якщо двигун не заведеться, то автомобіль не поїде.

Запишіть ці правила у вигляді продукції.

Розв'язання. Указані правила у вигляді продукції R1, R2, R3, R4 записані на мові оболонці для створення експертних систем "GURU".

```
RULESET: CAR
GOAL: MOVE

INITIAL: CLEAR
E.LSTR = 80
MOVE = UNKNOWN
AKKUM = UNKNOWN
STARTER = UNKNOWN
MOTOR = UNKNOWN
OUTPUT "Система діагностики автомобіля"

VARIABLE: MOVE
LABEL: Чи рухатиметься автомобіль?
VARIABLE: AKKUM
LABEL: Чи заряджений акумулятор?
FIND: input akkum using "u" with "Заряджений у Вас акумулятор (Y/N)?"
VARIABLE: STARTER
LABEL: Чи справний стартер?
FIND: input starter using "u" with "Справний у Вас стартер (Y/N)?"
VARIABLE: MOTOR
LABEL: Чи працює мотор?

DO:
CLEAR
OUTPUT "Автомобіль", MOVE

RULE: R1
IF: AKKUM <> "Y" OR STARTER <> "Y" THEN: MOTOR = 0
REASON: Якщо акумулятор сів або стартер не працює, то мотор не заведеться.

RULE: R2
IF: AKKUM = "Y" AND STARTER = "Y" THEN: MOTOR = 1
REASON: якщо акумулятор заряджений і стартер працює, то мотор заведеться.

RULE: R3
IF: MOTOP = 1 THEN: MOVE = "ПОЇДЕ"
REASON: Якщо мотор заведеться, то автомобіль поїде.

RULE: R4
IF: MOTOR = 0 THEN MOVE = "НЕ ПОЇДЕ"
REASON: Якщо мотор не заведеться, то автомобіль не поїде.
```

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

1. Сформулюйте і запишіть правила, які повторюють хід ваших думок під час переходу через дорогу, у вигляді продукції. Спочатку ви визначаєте, який колір на світлофорі. Якщо червоний – то чекаєте, якщо зелений - дивитеся, чи немає якого-небудь "божевільного" водія, який міг би їхати на червоний колір. Якщо є, то чекайте, поки він проїде. Якщо немає, то переходите через дорогу.

2. Сформулюйте і запишіть правила, що визначають несправність магнітофона, у вигляді продукції. Ви вмикаєте магнітофон, і він працює, але звучання погане. Ви перевіряєте, чи забруднена голівка. Якщо так, то необхідно протерти голівку. Якщо ні, то перевіряйте, чи правильно встановлена касета. Якщо ні, то поправляєте, а якщо правильно, то кажете, що необхідно викликати майстра.

3. Сформулюйте і запишіть правила, що визначають приймати чи ні людини на роботу, у вигляді продукції. Якщо людина не має вищої освіти, то відмовити. Якщо має, то скільки років пропрацював претендент за фахом. Якщо менше року, то відмовити, якщо більше - то прийняти. Якщо претендент має вчене звання, то запропонувати посаду наукового співробітника, якщо немає, то посада інженера-конструктора.

4. Сформулюйте і запишіть правила, які дають поради при створенні і відладці програми, у вигляді продукції. Якщо ви написали програму на мові високого рівня і при компіляції виявили синтаксичні помилки, то необхідно виправити програму. Якщо помилок немає, то необхідно запустити редактора зв'язків. Якщо редактор зв'язків повідомляє про помилки, то необхідно перевірити наявність всіх початкових модулів. Якщо помилок при лінкувані немає, то програма готова до роботи.

5. Сформулюйте і запишіть правила, які визначають, чи буде сьогодні дощ, у вигляді продукції. Спочатку ви визначаєте, чи ясне небо. Якщо небо ясне, то дощу не буде. Якщо небо похмуре, то ви дивитеся, чи є на небі чорні грозові хмари. Якщо немає, то дощу не буде. Якщо є, то дивитеся, в яку сторону вони рухаються. Якщо у вашу сторону, то дощ буде. Якщо ж ні - то не буде.

6. Сформулюйте і запишіть правила, що визначають, чи є у дитини труднощі при вивченні арифметики, у вигляді продукції. Якщо у дитини проблеми при вивченні складання, то вона зазнає труднощі при вивченні арифметики. Якщо дитина має проблеми при вивченні множення, то вона зазнає труднощі при вивченні арифметики. Аналогічно з відніманням і діленням.

7. Ви бажаєте прогнозувати на біржі рівень цін. Якщо валютний курс долара падає, то процентні ставки зростають. Якщо валютний курс долара зростає, то процентні ставки падають. Якщо процентні ставки зростуть, то рівень цін на біржі упадає. Якщо процентні стави упадають, то рівень цін на біржі зросте. Сформулюйте і запишіть ці правила у вигляді продукції.

8. Ми бажаємо визначити, чи буде в результаті весняного паводку повінь чи ні. Якщо рівень води в річці у межі міста високий і йдуть сильні дощі, тепла погода і багато снігу в горах, то очікується повінь. Якщо ж хоч би один з цих чинників не виконується, то повені не буде. Сформулюйте і запишіть ці правила у вигляді продукції.

9. Необхідно визначити, чи є даний об'єкт танком або автомобілем. У танка є гармата і люк. У автомобіля є дверці і колеса. У танка і автомобіля є кузов. Уточнюючи всі ці характеристики, ми повинні визначити об'єкт. Сформулюйте і запишіть ці правила у вигляді продукції.

10. Ви включаєте телевізор, а він не працює. Ви бажаєте визначити, чому це трапилося. Якщо запобіжник згорів, то його необхідно замінити. Якщо запобіжник цілий, то перевіряєте кабель живлення. Якщо він розірваний в якому-небудь місці, то необхідно його виправити. Якщо кабель живлення цілий, і ви самі розбираєтеся в радіоелектроніці, то лагодите телевізор. Якщо не розбираєтеся, то викликаєте майстра. Сформулюйте і запишіть ці правила у вигляді продукції.

ТЕМА 13. Нечітке подання знань

Поняття недостовірної та нечіткої інформація. Інтуїтивне поняття нечіткості. Об'єктивна та суб'єктивна невизначеність. Модальна логіка. Тризначна логіка Лукашевича. Логіка знання. Основи теорії можливостей. Логічне виведення за недостовірних знань. Неточне логічне виведення. Принципи неточного виведення. Логічне виведення за нечіткої інформації. Функція належності та основні операції над нечіткими множинам

ПРАКТИЧНЕ ЗАНЯТТЯ №12. НЕЧІТКЕ ПОДАННЯ ЗНАНЬ.

Мета: засвоїти поняття нечіткої множини та її функції належності, сформувати навички оперування цими поняттями.

План заняття:

1. Формалізування недостовірних та нечітких знань.
2. Логічні операції над нечіткими множинами.
3. Побудова та обчислення функції належності для нечітких множин.

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 13.

ПРИКЛАДИ РОЗВ'ЯЗАННЯ ТИПОВИХ ЗАДАЧ

Приклад 1. Нехай нечіткі множини A, B, C задані функціями належності:

$$A = 0{,}4/x_1 + 0{,}2/x_2 + 0/x_3 + 1/x_4,$$

$$B = 0{,}7/x_1 + 0{,}9/x_2 + 0{,}1/x_3 + 1/x_4,$$

$$C = 0{,}1/x_1 + 1/x_2 + 0{,}2/x_3 + 0{,}9/x_4.$$

Виконати логічні операції над цими множинами.

Розв'язання.

1) $A \subset B$, тобто A міститься в B або B домінує A; C непорівняно ні з A, ні з B, тобто пари $\{A, C\}$, $\{B, C\}$ - це пари недомінуючих нечітких множин.

2) $A \neq B \neq C$.

3) $\overline{A} = 0{,}6/x_1 + 0{,}8/x_2 + 1/x_3 + 0/x_4$, $\overline{B} = 0{,}3/x_1 + 0{,}1/x_2 + 0{,}9/x_3 + 0/x_4$.

4) $A \cap B = 0{,}4/x_1 + 0{,}2/x_2 + 0/x_3 + 1/x_4$.

5) $A \cup B = 0{,}7/x_1 + 0{,}9/x_2 + 0{,}1/x_3 + 1/x_4$.

6) $A - B = A \cap \overline{B} = 0{,}3/x_1 + 0{,}1/x_2 + 0/x_3 + 0/x_4$.

$B - A = \overline{A} \cap B = 0{,}6/x_1 + 0{,}8/x_2 + 0{,}1/x_3 + 0/x_4$

7) $A \oplus B = 0{,}6/x_1 + 0{,}8/x_2 + 0{,}1/x_3 + 0/x_4$.

Приклад 2. Нехай експерт визначає товщину виробу за допомогою понять "Мала товщина", "Середня товщина" і "Велика товщина", при цьому мінімальна товщина дорівнює 10 мм, а максимальна – 80 мм. Формалізувати поняття товщини виробу.

Розв'язання. Формалізування такого опису можна здійснити за допомогою лінгвістичної змінної $\langle \beta, T, X, G, M \rangle$, де:

β - товщина виробу;

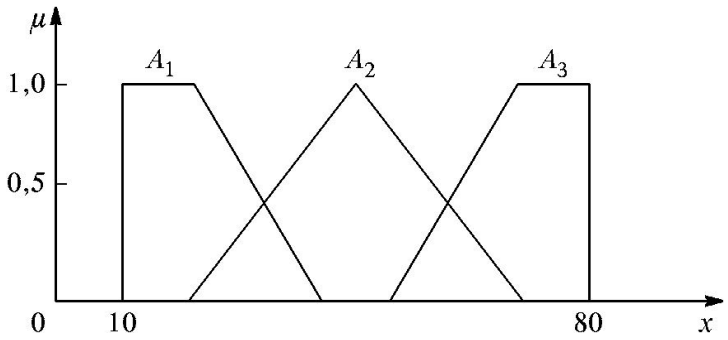
T - {"Мала товщина", "Середня товщина", "Велика товщина"};

X - [10, 80];

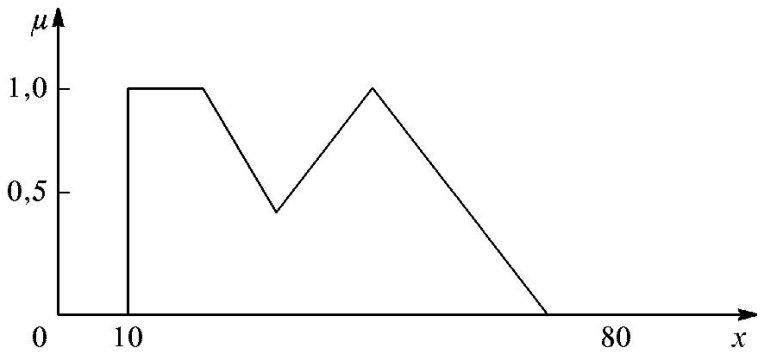
G - процедура утворення нових термів за допомогою зв'язок "і", "або" і модифікаторів типу "дуже", "не", "трішки" і т.п. Наприклад, "Мала або середня товщина", "Дуже мала товщина" і т.п.;

M - процедура задання на X = [10, 80] нечітких підмножин A_1 = "Мала товщина", A_2 = "Середня товщина", A_3 = "Велика товщина", а також нечітких множин для термів з $G(T)$ у відповідності з правилами трансляції нечітких зв'язок і модифікаторів "і", "або", "дуже", "не", "трішки" та інших операцій над нечіткими множинами виду: $A \cap B$, $A \cup B$, $\overline{A}$, $CON\ A = A^2$, $DIL\ A = A^{0,5}$ і т.п.

Терм-множину і розширений терм-множину в умовах прикладу можна охарактеризувати функціями належності, наведеними на мал. 1 і 2.



Мал.1. Функція належності нечітких множин A_1 = "Мала товщина", A_2 = "Середня товщина", A_3 = "Велика товщина".



Мал. 2. Функція належності нечіткої множини
$A_1 \cup A_2$ = "Мала або середня товщина".

ЗАДАЧІ ДЛЯ САМОКОНТРОЛЮ

1. Нехай нечіткі множини A, B, C задані функціями належності:

$A = 0{,}4/x_1 + 0{,}2/x_2 + 0/x_3 + 1/x_4$,

$B = 0{,}7/x_1 + 0{,}9/x_2 + 0{,}1/x_3 + 1/x_4$,

$C = 0{,}1/x_1 + 1/x_2 + 0{,}2/x_3 + 0{,}9/x_4$.

Виконати логічні операції над цими множинами.

2. Нехай нечіткі множини A, B, C задані функціями належності:

$A = 0{,}2/x_1 + 0{,}3/x_2 + 0{,}1/x_3 + 0{,}5/x_4$,

$B = 0{,}5/x_1 + 0{,}7/x_2 + 0{,}2/x_3 + 1/x_4$,

$C = 0{,}3/x_1 + 0{,}1/x_2 + 0{,}8/x_3 + 0{,}9/x_4$.

Виконати логічні операції над цими множинами.

3. Нехай нечіткі множини A, B, C задані функціями належності:

$A = 0{,}1/x_1 + 0{,}5/x_2 + 0{,}8/x_3 + 0{,}1/x_4$,

$B = 0{,}6/x_1 + 0{,}4/x_2 + 0{,}5/x_3 + 0{,}6/x_4$,

$C = 0{,}5/x_1 + 0{,}2/x_2 + 0{,}3/x_3 + 0{,}4/x_4$.

Виконати логічні операції над цими множинами.

4. Нехай нечіткі множини A, B, C задані функціями належності:

$A = 0{,}3/x_1 + 0{,}4/x_2 + 0{,}7/x_3 + 0{,}3/x_4$,

$B = 0{,}8/x_1 + 0{,}1/x_2 + 0{,}5/x_3 + 1/x_4$,

$C = 0{,}3/x_1 + 0{,}3/x_2 + 0{,}2/x_3 + 0{,}9/x_4$.

Виконати логічні операції над цими множинами.

5. Нехай нечіткі множини A, B, C задані функціями належності:

$A = 0{,}6/x_1 + 0{,}1/x_2 + 0/x_3 + 0{,}7/x_4$,

$B = 0{,}4/x_1 + 0{,}5/x_2 + 0{,}2/x_3 + 1/x_4$,

$C = 0{,}2/x_1 + 1/x_2 + 0{,}6/x_3 + 0{,}1/x_4$.

Виконати логічні операції над цими множинами.

6. Нехай нечіткі множини A, B, C задані функціями належності:

$A = 0{,}6/x_1 + 0{,}1/x_2 + 0{,}5/x_3 + 1/x_4$,

$B = 0{,}2/x_1 + 0{,}5/x_2 + 0{,}7/x_3 + 1/x_4$,

$C = 0{,}5/x_1 + 0{,}3/x_2 + 0{,}2/x_3 + 0{,}6/x_4$.

Виконати логічні операції над цими множинами.

7. Нехай експерт визначає вагу виробу за допомогою понять "Мала вага", "Середня вага" і "Велика вага", при цьому мінімальна вага дорівнює 1 кг, а максимальна – 10 кг. Формалізувати поняття ваги виробу.

8. Нехай експерт визначає висоту виробу за допомогою понять "Мала висота", "Середня висота" і "Велика висота", при цьому мінімальна висота дорівнює 1 м, а максимальна – 5 м. Формалізувати поняття висоти виробу.

9. Нехай експерт визначає довжину виробу за допомогою понять "Мала довжина", "Середня довжина" і "Велика довжина", при цьому мінімальна довжина дорівнює 20 см, а максимальна – 65 см. Формалізувати поняття довжини виробу.

10. Нехай експерт визначає вартість виробу за допомогою понять "Мала вартість", "Середня вартість" і "Велика вартість", при цьому мінімальна вартість дорівнює 150 грн., а максимальна – 2000 грн. Формалізувати поняття вартість виробу.

11. Нехай експерт визначає термін придатності виробу за допомогою понять "Малий термін придатності", "Середній термін придатності" і "Великий термін придатності", при цьому мінімальний термін придатності дорівнює 1 рік, а максимальний – 15 років. Формалізувати поняття термін придатності виробу.

12. Нехай експерт визначає трудомісткість виробу за допомогою понять "Мала трудомісткість", "Середня трудомісткість" і "Велика трудомісткість", при цьому мінімальна трудомісткість дорівнює 120 роб. год., а максимальна – 560 роб. год. Формалізувати поняття трудомісткість виробу.

ТЕМА 14. Поняття та загальна характеристика експертної системи (ЕС)

Сфера застосування ЕС. Структура експертної системи та її основні компоненти (вирішувач (інтерпретатор), робоча пам'ять (база даних), база знань, компоненти придбання знань, пояснювальний компонент, діалоговий компонент). Статичні та динамічні ЕС. Етапи розробки ЕС (ідентифікація, концептуалізація, формалізація, виконання, тестування, дослідна експлуатація).

ПРАКТИЧНЕ ЗАНЯТТЯ №13. ПОНЯТТЯ ТА ЗАГАЛЬНА ХАРАКТЕРИСТИКА ЕКСПЕРТНОЇ СИСТЕМИ.

Мета – сформувати навички роботи з оболонкою ЕС "GURU" для створення експертних систем.

План заняття:

1. Загальна характеристика оболонки ЕС "GURU".
2. Порядок створення експертної системи.
3. Створення експертних систем за допомогою оболонки "GURU".

Теоретичні відомості, що необхідні для виконання практичного завдання

Під експертною системою розуміється система, об'єднуюча можливості комп'ютера із знаннями і досвідом експерта так, що система може запропонувати розумну пораду або знайти розумне рішення поставленої задачі. Додатковою можливістю системи є здатність пояснити хід своїх міркувань в зрозумілій формі.

При створенні експертних систем на якій-небудь мові високого рівня програміст стикається з тим, що розробка інтерфейсу програми, реалізація її системних функцій вимагають великих витрат часу, ніж створення самого набору правил експертної системи (ЕС). Для того, щоб розвантажити розробника ЕС від такої роботи, існують спеціальні інструментальні засоби (оболонки) експертних систем. Такі інструментальні засоби є в ЕС MYCIN, GURU, LEONARDO, DENDRAL і ін.

Оболонка ЕС "GURU" має три режими роботи:

діалоговий: в ході діалогу типу "запит-відповідь" за допомогою розвиненої системи меню, не вдаючись до написання власних програм, користувач створює експертну систему;

природна мова: користувач на запит системи вводить фрази на природній мові і одержує результати. Наприклад, система питає "Ваш запит?". Написавши в командному рядку фразу "Знайти всіх, що працюють, 1977 року народження", користувач одержує від системи розумну відповідь;

командний: як в мовах високого рівня (МВУ), пишеться програма, компілюється і працює відповідно до ваших вимог.

Зазвичай застосовуються змішані режими.

Характеристики оболонки ЕС "GURU"

Основними характеристиками є: **інтерфейс користувача**, **машина логічних висновків і експертизи**, що зберігаються.

Інтерфейс користувача описує відносини між користувачем і системою. Користувач ставить завдання, а машина повинна її виконати або пояснити, чому не можна її виконати.

Машина логічних висновків - це програмне забезпечення (ПО), яке можна використовувати при рішенні задач шляхів аргументування.

Експертизи, що зберігаються, - це **набір правил**, що відображають **знання**. У кожному правилі є **посилка** (IF) і **висновок** (THEN).

Якщо машина логічних висновків визнає посилку вірною, то і висновок буде вірним.

Знаходячись в будь-якому меню, можна одержати підказку к діям, допустимим в цьому меню. Для цього викликається допомога одночасним натисненням <Ctrl-L>.

Правила "GURU"

Система "GURU" базується на правилах. Правило складається з посилки (IF) і висновку (THEN). Посилка може включати:

- різні типи і види змінних, підтримуваних "GURU";
- логічні оператори (EQ, NE, GT, GE, LT, LE, IN, AND, OR, XOR, NOT);
- числові оператори (+, - /, *, **);
- числові функції (SIN, COS і т.п.);
- символьні функції.

Висновок може включати команди:

- надання значення змінним;
- що дозволяють проконсультуватися з іншим набором правил;
- інші команди "GURU".

Правила зберігаються в звичайному текстовому файлі.

Приклад:

```
RULESET: EASYCALC
GOAL: INTRATE
RULE: R1
IF: MONTHPAY < 50
      THEN: PERIOD = 120
RULE: R2
IF: PERIOD > 90
      THEN: INTRATE =12.5
```

Тут EASYCALC - ім'я набору правил (RULESET указувати не обов'язково); INTRATE - ім'я змінної мети; R1, R2 - імена правил; PERIOD, INTRATE, MONTHPAY - змінні.

Стратегії управління

Одне з важливих питань для ЕС – це яке правило розглядати наступним. Цим процесом управляє машина логічних висновків.

При виборі правила користуються двома основними стратегіями управління: *прямим і зворотним висновками*.

Прямий висновок

Даний метод діє від посилки до висновку до тих пір, поки змінній не буде привласнено значення. Машина логічних висновків (MLV) починає проглядати набір правил спочатку і проводить переглядання його до тих пір, поки змінній не буде привласнено значення.

Спочатку шукається перше правило, в якому визначено і істинно значення посилки. Це правило буде виконано і отриманий результат можна буде використовувати для тестування інших правил. Далі система шукає наступне правило з певним і істинним значенням посилки. Це продовжується до тих пір, поки не буде виконано правило для змінної мети.

Приклад:
RULESET: EASYCALC
GOAL: INTRATE
RULE: R1
IF: RERIOD > 90
    THEN: INTRATE = 12.5
RULE: R2
IF: MOMTHPAY < 50
    THEN: PERIOD = 120
RULE: R3
IF: MOHTHPAY > 50
    THEN: PERIOD = 60
RULE: R4
IF: PERIOD < 90
    THEN: INTRATE = 11.0

Хай спочатку змінній MONTHPAY привласнено значення 42. MLV шукає в наборі правил те правило, де визначено і істинно значення посилки (це R2). Тоді змінній PERIOD привласнюється значення 120. Услід за тим, починаючи знову з першого правила, шукається правило, в якому визначено і істинно значення посилки (це R1). Змінній меті привласнюється значення 12.5. Мета досягнута, система закінчила роботу.

Зворотний висновок

Цей висновок - найбільш часто використовуваний метод управління. При цьому MLV починає з мети і, проглядаючи набір правил, знаходить перше правило, за допомогою якого можна досягти мети.

Якщо посилка цього правила визначена і вірна, то система виконує відповідні дії. Якщо посилка невизначна, то MLV тимчасово міняє мету -

встановлює як мету змінну, яка дозволить визначити істинність першої знайденої посилки, і шукає перше правило, визначне і вірне для нової поставленої мети.

Скористаємося попереднім прикладом. У цій ЕС мета (GOAL) - знайти INTRATE. Шукаємо перше правило, в якому обчислювалася б змінна мети (це R1). Але його не можна виконати, поки невідома PERIOD. Шукаємо правило, де знаходиться PERIOD (це R2). Припустимо, що MONTHPAY задане і рівне 42. Тоді виконується R2 і потім R1. Мета досягнута.

Але тепер припустимо, що MONTHPAY = 70. Тоді ланцюжок R2 - R1 не приводить до знаходження мети (PERIOD не визначено, отже, не визначено в цьому ланцюжку і INTRATE).

Починаємо спочатку і шукаємо наступне правило, де знаходиться мета INTRATE (це R4). Тепер необхідно визначити PERIOD (нову змінну мети). PERIOD знаходиться в правилі R3. Оскільки MONTHPAY = 70, то R3 - вірно, тоді PERIOD = 60. Далі, перевіряється R4. Воно вірне. Отже, INTRATE = 11.0.

Змінні

Робочі змінні.

Робоча змінна (РЗ) - це звичайна змінна.

Спочатку всі РЗ мають значення UNKNOWN. Їм можна привласнити значення будь-чого:

A = 12.5 - приклад числової змінної;

B = "це строкова змінна" - приклад строкової змінної;

C = TRUE

D - FALSE - логічні змінні.

Заздалегідь визначені змінні.

Існує два типу заздалегідь визначених змінних (ЗВЗ): середовища і утиліти. Середовище "GURU" визначається змінними середовища. Вони визначають різні функціональні характеристики середовища "GURU". Ім'я цієї змінної завжди починається з букви E. Наприклад: E.HELP = TRUE

Задавання ЗВЗ примушує "GURU" автоматично реагувати на помилку в команді, наприклад, E.LSTR = 80 (максимальна довжина символьного рядка дорівнює 80).

Змінні типу утиліти служать для різних допоміжних цілей. Вони починаються із знаку #. Наприклад:

#GOAL = INTRATE

#COAL визначає мету ЕС.

Вирази із змінними.

Числові: 2 + 4, DEPTH=5, 6 + DEPTH, 2**2, SQRT(4), 60/5, 5.67*PI.

Рядкові: NAME = "Іванов" + "Іван" - зчеплення NAME стає рівним "Іванов Іван".

Логічні: A = B, A<>B, A <= B, A >= B.

IN - вводиться для позначення тотожності одного елементу іншому (перевірка того, чи відповідають права і ліва частини виразу); допустимо,

треба знайти службовця, чиє прізвище Мінев або Манев або Монев, тоді вводимо логічний вираз: NAME IN [M$HEB].

Складені логічні вирази, наприклад:

15 > 9 AND 20 < 100 - істинний вираз;

9 > 15 AND 20 < 100 - помилковий вираз.

Пояснення та аргументування

Важливою характеристикою "GURU" є можливість пояснити весь хід дій при консультації з набором правил. Це робиться за допомогою команд HOW і WHY. Як це робити, буде пояснено надалі.

Синтаксис правил "GURU"

Запишемо приклад згідно синтаксичним правилам "GURU" і детально пояснимо його. Припустимо, ми проводимо технічний огляд автомобіля і бажаємо знати, поїде він чи ні. Ми перевіряємо акумулятор і стартер і на основі результатів огляду ухвалюємо рішення. Ось наші правила в цій експертній системі:

Правило 1. Якщо акумулятор сів або несправний стартер, то двигун не заведеться.

Правило 2. Якщо акумулятор заряджений і справний стартер, то двигун заведеться.

Правило 3. Якщо двигун заведеться, то автомобіль поїде.

Правило 4. Якщо двигун не заведеться, то автомобіль не поїде.

Запишемо ці правила ЕС з урахуванням синтаксису "GURU".

```
RULESET: CAR
GOAL: MOVE

INITIAL: CLEAR
E.LSTR = 80
MOVE = UNKNOWN
AKKUM = UNKNOWN
STARTER = UNKNOWN
MOTOR = UNKNOWN
OUTPUT "Система діагностики автомобіля"

VARIABLE: MOVE
LABEL: Чи рухатиметься автомобіль?
VARIABLE: AKKUM
LABEL: Чи заряджений акумулятор?
FIND: input akkum using "u" with "Заряджений у Вас акумулятор (Y/N)?"
VARIABLE: STARTER
```

LABEL: Чи справний стартер?
FIND: input starter using "u" with "Справний у Вас стартер (Y/N)?"
VARIABLE: MOTOR
LABEL: Чи працює мотор?

DO:
CLEAR
OUTPUT "Автомобіль", MOVE

RULE: R1
IF: AKKUM <> "Y" OR STARTER <> "Y"
THEN: MOTOR = 0
REASON: Якщо акумулятор сів або стартер не працює, то мотор не заведеться.

RULE: R2
IF: AKKUM = "Y" AND STARTER = "Y"
THEN: MOTOR = 1
REASON: Якщо акумулятор заряджений і стартер працює, то мотор заведеться.

RULE: R3
IF: MOTOP = 1
THEN: MOVE = "ПОЇДЕ"
REASON: Якщо мотор працює, то автомобіль поїде

RULE: R4
IF: MOTOR = 0
THEN MOVE = "НЕ ПОЇДЕ"

Пояснемо все більш детально.
RULESET: CAR - це ім'я набору правил (необов'язково указувати);
INITIAL: - це розділ ініціалізації. Сюди входять ті команди, які повинні виконатися до консультації з наборів правил.
CLEAR - команда для очищення екрану.
E.LSTR = 80 - ця змінна встановлює максимальну довжину символьних рядків.
MOVE = UNKNOWN
MOTOR = UNKNOWN - ініціалізація змінних, при якій їм привласнюється значення UNKNOWN.
ODTPUT "Система діагностики автомобіля" - виводить на екран символьний рядок.
VARIABLE: MOVE - визначається змінна, використовувана в наборі правил. Всі змінні повинні бути визначені.
LABEL: - Пояснення на природній мові, навіщо потрібна дана змінна.

FIND: - якщо в посилці зустрічається змінна з невизначеним значенням (UNKNOWN), яка не присутня в висновку якого-небудь правила, то виводяться ті команди, які знаходяться після FIND. Тут знаходиться команда введення: input akkum using "u" with "Заряджений у Вас акумулятор?". Ця команда чекає введення з екрану в змінну akkum символів "Y" або "N". У цій команді введення:

using "u" - шаблон введення;

with "..." - виводяться на екран у вигляді запиту для підказки.

RULE: R1 - ім'я правила;

IF: - посилка правила;

THEN: - висновок правила;

REASON - пояснення на природній мові, що робить правило.

DO - розділ завершення. Виконуються команди, які необхідні для виконання консультації з ЕС.

OUTPUT "Автомобіль", MOVE - виводиться рядок "Автомобіль" і услід за нею змінна MOVE.

(*Примітка*: опис всіх команд дане в розділі самостійної роботи.)

Система працює таким чином: послідовним перебором правил, починаючи з першого, знаходиться правило R3, що містить у висновку змінну мети MOVE. МЛВ визначає, що вона UNKNOWN, отже необхідно знайти умову для її знаходження. У правилі R3 ця умова задається змінною MOTOR. MOTOR – це змінна з невідомим значенням, що міститься в посилці правила R3, яка стає новою змінною ціллю. Ця змінна при послідовному переборі правил, починаючи з першого, вперше зустрічається в висновку правила R1. У посилці цього правила - дві змінних. Вони теж невідомі, причому їх не можна виявити в висновку якого-небудь правила, але вони описані в FIND. Отже, вводиться запит на введення цих змінних. Коли AKKUM і STARTER введені, то визначається MOTOR. Потім перевіряємо MOTOR і визначуваний MOVE. Мета досягнута.

Відладка КС

Запит під час консультації.

Під час консультації може створитися враження, що дії, що виконуються машиною логічних висновків, не мають відношення до проблеми. Це можливо тому, що користувач не знає, як відбувається внутрішній процес аргументування. Якщо користувач дійсно не розуміє, чому від нього вимагають тієї або іншої інформації, він може відреагувати, використовуючи Y (CTRL-Y). В цьому випадку він побачить на екрані дисплея поточне оброблюване правило. Після натиснення ENTER це пояснення зникає, і він може ввести відповідь.

Запит після консультації.

Після консультації з набором правил, користувач може попросити систему пояснити, які правила і змінні використовувалися. Для цього застосовуються дві команди:

HOW - видає змінні, які використовувалися;

WHY - пояснює правила, які використовувалися.

З'ясуємо, яку інформацію дає їх використання.

HOW - видає значення зміною мети, правило або правила, за допомогою яких була визначена мета.

HOW "ім'я змінної" - видає значення або значення з вказаною змінною.

HOW "число" - видає значення або значення змінної з порядковим номером, заданими цією змінною в наборі правил.

WHY - відтворює на екрані дисплея пояснення (REASON) і змінні, які були потрібні для правила, що виконується останнім. Змінні відображаються з порядковий номером, який можна потім використовувати в команді HOW.

WHY "ім'я правила" - відтворює на екрані дисплея пояснення (REASON) і змінні, необхідне для даного правила.

WHY "число" - відтворює на екрані дисплея пояснення і змінні, необхідне для REASON правила з вказаним порядковим номером в наборі правил.

Для того, щоб пояснити процес аргументування, необхідно використовувати HOW і WHY спільно.

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке прямій і зворотний висновок? Назвіть основні відмінності між ними?
2. З яких частин складається правило?
3. Типи змінних і їх особливості.
4. Скільки інтерфейсів має "GURU"?
5. Що роблять команди INPUT і OUTPUT? Їх особливості.
6. Структура оболонки ЕС "GURU"?
7. Синтаксис правил "GURU".

ПІДГОТОВКА ДО ВИКОНАННЯ ЗАВДАННЯ

1. Ознайомтеся з прикладом розв’язання типового завдання.
2. Дайте відповіді на контрольні питання.
3. Ознайомтеся із готовим варіантом ЕС.
4. Підготуйте виправлення в заданому варіанті ЕС для внесення змін в ході виконання завдання.

ПРИКЛАД РОЗВ’ЯЗАННЯ ТИПОВОГО ЗАВДАННЯ

1. Запустіть "GURU" з командного рядка.
2. Система виводить на екран: ІМ'Я НОВОГО СЕАНСУ...

Введіть ім'я вашого сеансу роботи. Це ім'я в подальшому використовується для завантаження сеансу вашої роботи з "GURU".

3. Виберіть в основному меню рядок "Експертні системи". У меню, що з'явиться, виберіть рядок "Створення експертної системи".

Якщо ви вводитимете весь набір правил, опис змінних, розділи ініціалізації і завершення наново, виберіть рядок "Нова база знань". З'являється підказка, що запрошує ім'я нового набору правил - бази знань (БЗ). Введіть ім'я нового набору правил. Далі з'явиться меню поточного набору правил "Будівник бази знань".

4. Якщо ви редагуватимете наявний набір правил, виберіть рядок "Існуюча база знань".

Нижчеописані дії орієнтовані на нову базу знань.

5. Виберіть рядок меню "'Визначення". Виберіть рядок меню "Мета". У рядку підказки введіть мету і натисніть "ВК". Виберіть рядок меню "Попереднє меню".

6. Виберіть рядок меню "Правила". Виберіть рядок меню "Створення". Введіть ім'я правила і натисніть <ВК>. Ви потрапляєте в середовище створення правила. Перехід від одного до іншого вікна - за допомогою клавіш PgDn і PgUp.

7. У полі IF введіть посилку. У полі THEN введіть висновок. У полі ВИВІД введіть пояснення. За допомогою клавіші <ESC> вийдіть з середовища створення правила.

8. Виберіть рядок меню "Редагування". Прогляньте введене правило. Виправить, якщо є, помилки.

9. Аналогічно введіть всі правила. Виберіть "Перегляд". Прогляньте всі введені правила. Виберіть "Попереднє меню" і ще раз "Попереднє меню".

10. Виберіть рядок меню "Змінні". Виберіть рядок "Створення". Ви потрапите в середовище для створення змінних. Перехід між вікнами - за допомогою клавіш PgDown і PgUp.

11. Введіть у вікно "LABEL" пояснення. Введіть у вікно "FIND" інформацію (якщо необхідно). Натисніть <ESC>.

12. Виберіть рядок меню "Редагування". Проглянете створену змінну. Введіть, якщо потрібно, виправлення.

13. Аналогічно створіть всі необхідні змінні. Виберіть рядок меню "Перегляд". Прогляньте всі створені змінні.

14. Виберіть двічі рядок "Попереднє меню".

15. Виберіть "Ініціалізація". На екрані з'явиться текстовий редактор будівника бази знань. Тут вводяться команди, які відпрацьовуються до того, як буде запущена ЕС. Натисніть <ESC>.

16. Виберіть рядок меню "Завершення". На екрані з'являється текстовий редактор бази знань. Тут вводяться команди, які виконуються

після того, як виконаний сеанс роботи з експертною системою. Натисніть <ESC>.

17. Виберете рядок меню "Вихід".

18. Виберіть рядок меню "Збереження". Ваш набір правил зберігається у файлі з розширенням *.rss.

19. Виберіть "Компіляцію". Відкомпілюйте ваш набір правил. Проглянете результати компіляції. Виправить базу знань відповідно до цих результатів і знову відкомпілюйте (якщо це буде потрібно). Для цього поверніться в попереднє меню і повторіть всі операції по пунктах 4-15, виключаючи ті, які вам не буде потрібно. Виберіть рядок меню "Кінець".

20. Виберіть рядок меню "Попереднє меню". Виберіть рядок "Консультація з експертною системою". Перевірте працездатність вашої експертної системи. Якщо вона не працює, то виправить набір правил.

21. Щоб перевірити, як виконувалася ваша ЕС, необхідне використання HOW і WHY.

Коли ви закінчите консультацію, виберіть рядок меню "Пояснення логічних висновків". Виберіть рядок меню "Пояснення мети". За умовчанням - це команда HOW без параметрів. На екрані ви побачите, яке правило використовувалося для визначення значення змінного ланцюга.

Після цього використовуйте WHY для того, щоб зрозуміти, чому використовувалося це правило. Якщо HOW показала, що використовувалося правило, наприклад, R10, то введіть WHY R10 для того, щоб розібратися, чому використовувалося це правило і які змінні це правило застосовувало.

Скористайтеся HOW знову для того, щоб з'ясувати, які правила використовувалися для отримання значень, представлених в таблиці змінних з останньої команди WHY.

Якщо з таблиці, призначеної для правила R10, стає відомо, що використовувалася змінна 8, введіть HOW 8 для того, щоб переглянути, яке правило дало змінній 8 її поточне значення.

Продовжуючи цей процес, можна зрозуміти, як ЕС дійшла зробленого нею висновку.

ЗВІТ ПРО ВИКОНАНЕ ЗАВДАННЯ ПОВИНЕН МІСТИТИ:

1. Відповіді на контрольні питання.
2. Виправлений варіант експертної системи.
3. Протоколи консультації з експертною системою при різних вхідних даних.
4. Протоколи пояснень експертної системи про правила і змінні, що використовувалися ЕС (команди HOW, WHY).
5. Висновки з роботи.

ЗАВДАННЯ ДЛЯ САМОКОНТРОЛЮ

Варіант 1
GOAL: ways

```
/* Цей набір правил дозволить вам одержати ряд порад по/
/* усуненню неполадок вашого улюбленого автомобіля і */
/* причин їх появи. Звичайно, це маленький і не повний */
/* набір, написаний людиною, що погано знає цю галузь, але */
/* він і призначений для показу */
/*засобів guru.*/
/* На питання системи слід вводити відповідне*/
/* значення булевої змінної (да - true, ні - false )*/

INITIAL:
Clear
release variable        /* прибираємо непотрібні нам змінні */
e.lstr=250    /* максимальна довжина рядка */
output "Добридень, МІСТЕР (MICIC)."
output
output "Ви, подорожуючи на своєму автомобілі, зупинилися"
output "віддихнути. А коли набралися сил, то виявили, що
output "ваша машина не заводиться. Ми постараємося дати поради"
output "по усуненню і причинам появи несправностей."
output "Але для цього ви повинні надати всю інформацію."
output "Отже, поїхали."
output
fires = true
input fires logic with "Чи є іскра в блоці запалення?"

DO:
clear
output "Ось що я вам скажу, люб'язний."
output
output reasons
output
output "А ось що вам слід зробити в даній ситуації."
output
output ways

RULE: Rl
IF: fires
       THEN: output
              input petrol logic with "Чи надходить бензин в карбюратор?"
REASON: якщо є іскра, то потрібен ще і бензин.
```

COMMENT: чи надходить бензин в карбюратор.

RULE: R2
IF: not fires
THEN: output
input acommulate logic with "Окислені клеми акумулятора?"
REASON: якщо немає іскри, то швидше за все окислилися клеми ак-ра.
COMMENT: чи окислювався акумулятор.

RULE: R3
IF: not petrol
THEN: output "Чи є бензин в баку вашого автомобіля?"
input ptrltank logic
REASON: Якщо бензин в карбюратор не поступає, то швидше за все, він просто закінчився.
COMMENT: Немає бензину в баку автомобіля.

RULE: R4
IF: not ptrltank
THEN: rasons = "Закінчився бензин в баку вашого автомобіля."
ways = "Заправте машину пальним."
REASON: Якщо немає бензину, то треба заправитися.
COMMENT: Порожній бак.

RULE: R5
IF: ptrltank
THEN: reasons="Засорилася трубка бензонасоса."
ways = "Від'єднайте трубку і продуйте. Потім, встановивши"
ways = ways + "на місце, спробуйте знову завести."
REASON: Якщо бензин в баку є, а в карбюраторі його немає, то треба прочистити трубки, що поставляють паливо в карбюратор.
COMMENT: засмітилися канали подачі бензину.

RULE: R6
IF: acommulate
THEN: reasons="Плохой контакт ланцюга запалення з ак-ром."
ways = "Зачистите клеми наждачною шкіркою і спробуйте"
ways = ways+"завести знову."
REASON: Якщо окислені контакти, то їх треба зачистити.
COMMENT: поганий контакт акумулятора з проводкою.

RULE: R7
IF: not acommulate
THEN: output "Акумулятор виробив свій ресурс?"
input lowenergy logic

REASON: Якщо немає іскри і контакт в порядку, то швидше за все ваш акумулятор став непридатним.
COMMENT: ресурс акумулятора.

RULE: R8
IF: lowenergy
THEN: reasons = "Ваш акумулятор став непридатним."
ways = "Якщо є можливість, то замініте свій"
ways = ways + "акумулятор. Інакше вам доведеться"
ways = ways + "заводити свій апарат 'ручкою' (зігнутий"
ways = ways + "шматок залізки)"
REASON: Якщо сів акумулятор, то його треба міняти або заводити машину 'ручкою'
COMMENT: сів акумулятор.

VAR: fires
LABEL: наявність іскри в блоці запалення

VAR: ways
FIND: reasons = "Причина появи цих неполадок невідома."
ways = "Попробуйте звернутися в автосервіс."
LABEL: спосіб усунення неполадок

END:

У варіанті 1 пропонується передбачити зміни, що враховують ситуації:
а) автомобіль заводиться, але не їде;
б) автомобіль заводиться, їде, але не туди, куди його направляє водій.

ТЕМА 15. Подання знань в експертній системі

Визначення складу та моделі представлення знань в ЕС. Фактори, що визначають склад знань ЕС: проблемне середовище, архітектура ЕС, потреби та цілі користувачів, мова спілкування. Знання, необхідні для ЕС (знання про процес розв'язування задачі, про мову спілкування та способах організації діалогу, про способи представлення та модифікації знань, підтримуючи структурні та управлінські знання, знання про методи взаємодії із зовнішнім середовищем, знання про модель зовнішнього світу). Інтерпретація знання.

ПРАКТИЧНЕ ЗАНЯТТЯ №14. ПОДАННЯ ЗНАНЬ В ЕС.

Мета: самостійне програмування у повному обсязі найпростішої експертної системи.

План заняття:

1. Формування набору правил та бази знань ЕС.
2. Програмна реалізація заданого варіанту ЕС.
3. Створення пробної ЕС засобами оболонки ЕС "GURU".

Теоретичні відомості, що необхідні для виконання даної роботи, містяться в конспекті лекцій за темою 15.

ПРИКЛАД РОЗВ'ЯЗАННЯ ТИПОВОГО ЗАВДАННЯ

Приклад експертної системи, що ілюструє можливості "GURU".

```
/* INVESTORS.RSS */
/* */

GOAL: advice

INITIAL:
e.tryp = "e"
e.lstr = 80 /* Максимальна довжина рядка 80 */
e.deci = 0 /* Немає цифр після десяткової коми */
e.lnum = 8 /* Довжина числа */
savperdep = 5000 /* Допустимий мінімум заощаджень */
/* з розрахунку на одного утриманця, при якому загальні */
/* заощадження сім'ї можна назвати "добрими" (true) */
incperdep = 4000 /* Мінімально допустимий дохід на */
/* одного утриманця */
basincome = 15000 /* Мінімально допустимий дохід */

/* розділи сім'ї */
/* Загальний дохід сім'ї визначається об'єднанням */
/* incperdep і basincome */
```

```
advice = unknown
goodsave = unknown
goodincome = unknown
income = unknown
savings = unknown
steady = unknown
needincome = unknown
dependents = unknown
newcash = unknown

clear
output “КАПІТАЛОВКЛАДЕННЯ”
input newcash num

DO: /* Цей розділ виконується після того */
/* як оброблені правила */

output “НА ОСНОВІ ДАНОЇ ІНФОРМАЦІЇ:”
test advice
case “АКЦІЇ”:
output “ВАМ СЛІД ВКЛАСТИ ВСЮ СУМУ В АКЦІЇ. ”
break
case “ЗАОЩАДЖЕННЯ”:
output “ВАМ СЛІД ПОМІСТИТИ ВСЮ СУМУ В ЗАОЩАДЖЕННЯ. ”
break
case “КОМПРОМІС”:
tosave = min (newcash, (savperdep * dependents) - savings)
tostock = max(0, newcash - tosave)
output “ВАМ СЛІД ПОМІСТИТИ В ЗАОЩАДЖЕННЯ ”
tosave using “$ff,fff,fff”
if tostock > 0 then
output “І ВКЛАСТИ  ”, tostock using “$ff,fff,fff” “  У АКЦІЇ.”
endif
break
endtest
e.deci = w
e.lnum = 14

RULE: R1
IF: goodincome and goodsave
        THEN: advice = “АКЦІЇ”
NEEDS: goodincome goodsave
REASON: вкладати в акції, якщо клієнт надійний у фінансовому
відношенні
```

RULE: R2
IF: not goodincome
THEN: advice = “ЗАОЩАДЖЕННЯ ”
NEEDS: goodincome
REASON: не вкладайте в акції, якщо ваш дохід в даний час нестійкий

RULE: R3
IF: not goodsave and goodincome
THEN: advice = “КОМПРОМІС ”
REASON: якщо заощадження невеликі, тоді вони повинні бути збільшені до того, як їх вкладати

RULE: R4
IF: not steady
THEN: goodincome = false
REASON: для хорошого доходу необхідна постійна робота

RULE: R5
IF: not (income > needincome)
THEN: goodincome = false
REASON: дохід не залежить від вас і від утриманців

RULE: R6
IF: not (income > needincome)
THEN: goodincome = true
REASON: щоб дохід був хороший, клієнт повинен мати постійну роботу

RULE: R7
IF: known(“income”) and known(“dependents”)
THEN: needincome = baseincome + (dependents * incperdep)
REASON: необхідний дохід - це дохід, який вам необхідний плюс загальний дохід всіх ваших утриманців

RULE: R8
IF: savings > (saveperdep * dependents)
THEN: goodsave = true
REASON: заощадження клієнтів повинні залежати від них самих і від утриманців

RULE: R9
IF: savings <= (saveperdep * dependents)
THEN: goodsave = false
REASON: заощадження клієнтів повинні залежати від них самих і від утриманців

NEEDS: savings dependents
REASON: заощадження клієнтів не залежать від них самих і від утриманців.

/* ВИЗНАЧЕННЯ ЗМІННИХ */

VAR: newcash
LABEL: сума готівки для внеску

VAR: advice
LABEL: дана порада

VAR: goodincome
LABEL: поточний дохід - добрий

VAR: goodsave
LABEL: поточні заощадження - добрі

VAR:needincome
LABEL: необхідна сума доходу

VAR: income
FIND: input income num with "ЯКИЙ ВАШ РІЧНИЙ ДОХІД СІМ'Ї?"
LABEL: поточний дохід

VAR: savings
FIND: input savings num with "СКІЛЬКИ У ВАС ЗАОЩАДЖЕНЬ?"
LABEL: поточні заощадження

VAR: steady
FIND:
output "ЧИ МОЖЕТЕ ВИ ОЧІКУВАТИ СТАБІЛЬНИЙ ДОХІД"
output "НА НАСТУПНИЙ РІК? (y/n)"
input steady str using "u"
steady = (steady = "Y")
LABEL: надійний дохід

VAR: dependents
FIND:
output ”СКІЛЬКИ У ВАС УТРИМАНЦІВ?"
input dependents num using "dd"
LABEL: число утриманців

END:

Опишемо детально роботу набору правил EC INVESTORS.

В INITIAL йде ініціалізація змінних. Розглянемо її окремі рядки.

e.tryp = 'e' - задає стратегію оцінки посилки (частини “if” правила), що містить невідомі змінні. Істинність посилки оцінюється відразу ж після того, як чергова невідома змінна стає відомою. Тестування посилки припиняється (не дивлячись на те, що всі змінні в ній ще не визначені), якщо тільки вдається безумовно встановити її істинність або помилковість.

e.lstr = 80 - максимальна довжина символьного рядка, який може виводитися на екран.

e.lnum = максимальна довжина числа.

В VAR описуються призначені для користувача змінні (див. практичну роботу №13).

Частина DO - закінчення роботи експертної системи. Конструкція test ... case ... endtest перевіряє змінну advice і залежно від її значення виконує ті або інші дії.

Розглянемо, як може працювати ця система.

Після запуску відбувається ініціалізація змінних. Консультація з ЕС йде методом зворотного аргументування. Система "GURU" в цьому випадку починає з кінця. Визначається мета ADVICE. Оскільки вона невідома, то продивляються ті правила, які визначають ADVICE. У нашому випадку першим правилом, де визначається ADVICE, буде R1. Тут невідомі GQODINCOME і GOODSAVE.

GQODINCOME спочатку визначається в R4, GOODSAVE - в R8. У R4 перевіряється змінна STEADY. Її значення запрошується за допомогою оператора FIND в описі змінної.

Якщо R4 не може бути виконане (посилка помилкова), то тестується R5, якщо R4 істинно (посилка вірна), змінною GOODINCOME привласнюється значення "false", "GURU" визначає, що посилка правила R1 невірна і переходить до оцінки посилки правила R2. У R2 посилка вірна, тому з цього правила визначається advice - мета системи. На цьому висновок завершується.

При тестуванні R5 нам доведеться визначити значення змінних INCOME і NEEDINCOME і т.д.

Аналогічно знаходиться змінна GOODSAVE (правила R8 і R9).

На рис.1 показана частина дерева рішень.

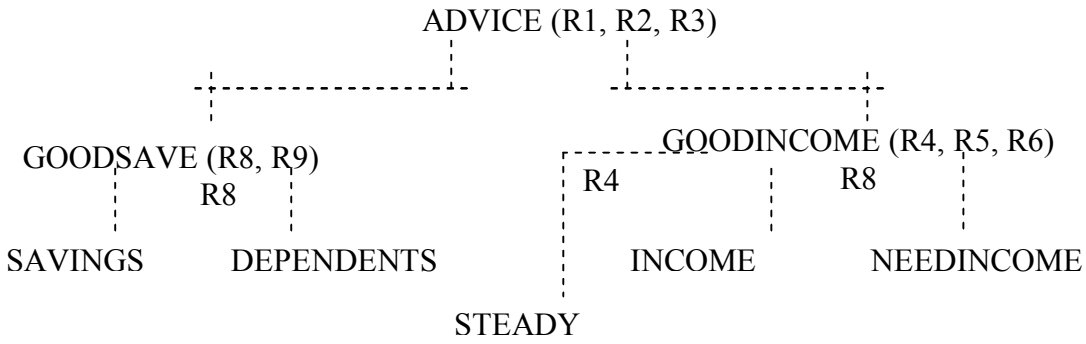


Рис. 1.

У дужках на рис. 1 показані правила, де визначається дана змінна, або FIND, якщо вона вводиться з клавіатури.

Перше питання, яке задасть ЕС: "Яку суму готівки Ви бажали б вкласти?"

Введіть: 12000 <Enter>

Наступне питання: "Чи можете Ви очікувати на стабільний дохід у наступному році?"

Введіть: “Y”

"Який Ваш річний дохід сім'ї?”

Введіть: 35000 <Enter>

"Скільки у Вас утриманців?"

Введіть: 4

"Скільки у Вас заощаджень?”

Введіть: 10000 <Enter>

Експертна система виводять суму, яку слід відкласти в заощадження і суму, яку необхідно вкласти в акція.

Розберіть детально роботу цієї ЕС для того, щоб створити свою.

КОНТРОЛЬНІ ПИТАННЯ

1. Перерахуйте і назвіть способи, за допомогою яких можна визначити або привласнити значення змінним в "GURU" (у діалоговому і програмному режимах).
2. Назвіть основні риси відмінності і схожість між мовою "GURU" і якою-небудь поширеною мовою високого рівня.
3. Приведіть алгоритм створення ЕС (основні кроки).
4. Вкажіть, змінні яких типів можна організувати в "GURU".

ПІДГОТОВКА ДО ВИКОНАННЯ ЗАВДАННЯ

1. Ознайомитеся з прикладом розв’язання типового завдання.
2. Відповісти на контрольні питання.
3. Відповідно до заданого варіанту написати програму, що реалізовує невелику експертну систему.

ПОРЯДОК ВИКОНАННЯ ЗАВДАННЯ

1. Пред'явите викладачу текст програмної реалізації заданого варіанту ЕС, написаний в ході домашньої підготовки.
2. Відладьте ЕС.
3. За допомогою HOW і WHY перевірте правильність роботи системи.
4. Покажіть результати роботи ЕС викладачу.

ЗВІТ ПРО ВИКОНАНЕ ЗАВДАННЯ ПОВИНЕН МІСТИТИ:

1. Відповіді на контрольні питання.
2. Варіант ЕС на природній мові і мові "Guru".
3. Пояснення, чому в написаній вами ЕС вибрані ті або інші змінні.
4. Протоколи консультації з експертною системою при різних вхідних даних.
5. Протоколи пояснень експертної системи про правила і змінні, що використовувалися ЕС (команди HOW, WHY).
6. Висновки з роботи.

ЗАВДАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Розробіть ЕС, яка повторює хід ваших думок під час переходу через дорогу. Спочатку ви визначаєте, який колір на світлофорі. Якщо червоний - чекаєте, якщо зелений - дивитеся, чи немає якого-небудь "божевільного" водія, який міг би їхати на червоний колір. Якщо є, то чекайте, поки він проїде. Якщо немає, то переходите через дорогу.

2. Створить ЕС, що визначає несправність магнітофона. Ви вмикаєте магнітофон, і він працює, але звучання погане. Ви перевіряєте, чи забруднена голівка. Якщо так, то необхідно протерти голівку. Якщо ні, то перевіряйте, чи правильно встановлена касета. Якщо ні, то поправляєте, а якщо правильно, то кажете, що необхідно викликати майстра.

3. Створіть ЕС, що визначає приймати чи ні людини на роботу. Якщо людина не має вищої освіти, то відмовити. Якщо має, то скільки років пропрацював претендент за фахом. Якщо менше року, то відмовити, якщо більше - то прийняти. Якщо претендент має вчене звання, то запропонувати посаду наукового співробітника, якщо немає - то посаду інженера-конструктора.

4. Розробіть ЕС, яка дає поради при створенні і відладці програми. Якщо ви написали програму на мові високого рівня і при компіляції виявили синтаксичні помилки, то необхідно виправити програму. Якщо помилок немає, то необхідно запустити редактора зв'язків. Якщо редактор зв'язків повідомляє про помилки, то необхідно перевірити наявність всіх початкових модулів. Якщо помилок при лінкувані немає, то програма готова до роботи.

5. Розробіть ЕС, яка визначає, чи буде сьогодні дощ. Спочатку ви визначаєте, чи ясне небо. Якщо небо ясне, то дощу не буде. Якщо небо

похмуре, то ви дивитеся. чи є на небі чорні грозові хмари. Якщо немає, то дощу не буде. Якщо є, то дивитеся, в яку сторону вони рухаються. Якщо у вашу сторону, то дощ буде. Якщо ж ні - то не буде.

6. Розробіть ЕС, що визначає чи є у дитини труднощі при вивченні арифметики. Якщо у дитини проблеми при вивченні складання, то вона зазнає труднощі при вивченні арифметики. Якщо дитина має проблеми при вивченні множення, то вона зазнає труднощі при вивченні арифметики. Аналогічно з відніманням і діленням.

7. Ви хочете прогнозувати на біржі рівень цін. Якщо валютний курс долара падає, то процентні ставки зростають. Якщо валютний курс долара зростає, то процентні ставки падають. Якщо процентні ставки зростуть, то рівень цін на біржі упаде. Якщо процентні ставки падають, то рівень цін на біржі зросте.

8. Ми бажаємо визначити, чи буде в результаті весняного паводку повінь чи ні. Якщо рівень води в річці у межі міста високий і йдуть сильні дощі, тепла погода і багато снігу в горах, то очікується повінь. Якщо ж хоч би один з цих чинників не виконується, то повені не буде.

9. Необхідно визначити, чи є даний об'єкт танком або автомобілем. У танка є гармата і люк. У автомобіля є дверці і колеса. У танка і автомобіля є кузов. Уточнюючи всі ці характеристики, ми повинні визначити об'єкт.

10. Ви включаєте телевізор, а він не працює. Ви бажаєте визначити, чому це трапилося. Якщо запобіжник згорів, то його необхідно замінити. Якщо запобіжник цілий, то перевіряєте кабель живлення. Якщо він розірваний в якому-небудь місці, то необхідно його виправити. Якщо кабель живлення цілий, і ви самі розбираєтеся в радіоелектроніці, то лагодите телевізор. Якщо не розбираєтеся, то викликаєте майстра.

ТЕМА 16. Характеристика бази знань експертних систем

Засоби для визначення релевантних знань. Зв'язність знання та даних, механізм доступу до знань та синтаксична, параметрична, семантична відповідність. Механізми пошуку знань в експертній системі (пошук в одному просторі, пошук в ієрархічних просторах, пошук при неточних і неповних даних тощо).

ПРАКТИЧНЕ ЗАНЯТТЯ №15. СТВОРЕННЯ ЕКСПЕРТНОЇ СИСТЕМИ ПРИ НЕТОЧНИХ І НЕПОВНИХ ДАНИХ.

Мета: сформувати навички застосування чинників упевненості та нечітких змінних при побудові ЕС з неточними і неповними даними.

План заняття:

1. Розглянути вбудовану алгебру обчислення ЧУ.
2. Розібрати приклад розв'язання типового завдання.
3. Створити базу знань на основі індивідуального варіанту ЕС.

Теоретичні відомості, що необхідні для виконання даної роботи

Нечітка логіка

Чинники упевненості використовуються у області математики, що називається нечіткою логікою. Оскільки евристичні правила "ЯКЩО - ТО" (тобто емпіричні, одержані з особистого досвіду експерта) ґрунтуються виключно на людському досвіді, то з повною визначеністю не можна сказати, що вони вірні. Користувач ЕС також не може бути повністю упевнений, що значення, які він надає змінним, абсолютно коректні.

Наприклад, правило:

Якщо процентні ставки зростають і податки зменшуються, то рівень цін на біржі зростає,

вірно не завжди, тому можна приписати йому значення деякого чинника упевненості (ЧУ). ЧУ може мати значення від 0 до 100. Звичайно, правила, для яких ЧУ = 0, розглядати немає сенсу. А якщо ЧУ = 100, то це повна упевненість в тому, що правило вірно.

Нехай приведене правило має ЧУ, рівний 90, і не можна стверджувати, що процентні ставки спадають, тобто першій умові правила призначене ЧУ, рівний 60. Крім того, припустимо, що податки коливаються (то збільшуються, то зменшуються), тому припустити зменшення податків можна тільки з ЧУ = 80. Тоді правило можна записати так: "Якщо процентні стави спадають (ЧУ = 60) і податки зменшуються (ЧУ = 80), то рівень цін на біржі зростає (ЧУ = 90). ЧУ, що рівень цін на біржі буде зростати, може бути підрахований, наприклад, таким чином: вибирається мінімальний ЧУ для умов частини "ЯКЩО" правила, розділених логічним оператором "І", і умножається на ЧУ для всього правила.

Для приведеного правила (minimum (60, 90) * 90) /100=54; поділений на 100 для нормалізації ЧУ). Отже, з ЧУ=54 можна сказати, що рівень цін на біржі спадатиме.

Можливі різні дії з ЧУ. Наприклад, можна було б так об'єднати ЧУ: maximum (minimum (60,80), 90) = 90. Тобто об'єднання ЧУ залежить від контексту поставленого завдання (але при цьому існують певні правила, які будуть пояснені нижче).

ЧУ задаються за допомогою покажчика cf, наприклад, problem = alternator cf 80. "GURU" дозволяє зв'язати ЧУ з будь-якою робочою змінною, використовуваною набором правил.

ЧУ використовуються для того, щоб визначити міру достовірності в наших знаннях. При накопиченні даних наша упевненість в окремому факті може або зростати, або зменшуватися.

Результати, одержані на основі застосування достовірних чинників, значно вагоміші за тих, які були одержані на основі застосування менш достовірної інформації. Якщо ЧУ не вказаний для якого-небудь значення, тоді передбачається, що cf = 100.

ЧУ використовують також для того, щоб визначити, чи відома конкретна змінна чи ні. Якщо значення ЧУ вище, ніж граничне значення, що визначається користувачем та встановлюється за допомогою змінного середовища E.UNKN, тоді вважається, що ця змінна відома, і що на основі даного факту можна робити висновки.

ЧУ можна явно задавати, використовуючи оператори "GURU". ЧУ можуть задаватися самим користувачем.

Вбудована алгебра обчислення ЧУ

Якщо в набір правил включена одна або декілька змінних, ЧУ яких нижче 100, то під час консультації "GURU" аргументуватиме з цими показниками вірогідності. Метод їх використання в процесі аргументування відносяться до алгебри ЧУ. Різним ситуаціям відповідає своя алгебра.

Для більшої гнучкості в системі "GURU" використовується широкий діапазон вбудованої алгебри обчислення ЧУ. В процесі будь-якої консультації ви можете самі вибирати наявну алгебру шляхом задавання потрібним змінним середовища необхідних значень.

Значення змінних середовища E.CFJO, E.CFCO, E.CFVA і E.ONKN указують, яким чином здійснюватиметься обчислення ЧУ системою "GURU" під час консультації. Необхідно відмітити, що дані змінні середовища не є "жорстко закріпленими" до набору правил. Це означає, що можна динамічно змінювати методи алгебри, не міняючи при цьому сам набір правил.

E.CFJO і E.CFCO контролює процес обчислення загального ЧУ логічного виразу, якщо розглядається посилка правила. Якщо одержаний в результаті ЧУ перевищує E.UNKN, то "GURU" вважає посилку відомою з достатньої ступенем упевненості, щоб включити правило в перелік допустимих при використанні. Якщо ЧУ посилки менше або рівний E.UNKN, то посилка вважається невідомою і правило не буде включено.

Пояснимо різницю у використанні E.CFJO і E.CFCO.

Припустимо, що вирази 1 і 2 істинні в деякій посилці кожне окремо (з чинниками упевненості, відповідно, CF1 і CF2). Причому CF1 і CF2

перевищують граничне значення ЧУ E.UNKN, при якому (або меншому) посилка з цими виразами була б недопустимою для використання. Нам необхідно тепер "визначити" міру достовірності посилок вигляду:

а) вираз 1 І вираз 2;
б) вираз 1 АБО вираз 2.

І "а" і "б" істинні, але міра достовірності "б" вище, ніж міра достовірності "а". Це пояснюється тим, що у випадку з "а" істинними повинні бути обидва вирази, а у випадку з "б" для того, щоб загальне значення було істинним необхідна істинність тільки одного виразу. E.CFJO задає міру достовірності у випадку "а", E.CFCO обробляє випадок "б".

Оцінка ЧУ всього правила (у припущенні, що були задані ЧУ посилки і висновку окремо), здійснюється з використанням тільки E.CFJO.

В тому випадку, якщо значення окремої змінної було набуте в результаті дії двох правил, для забезпечення зарахування ЧУ цієї змінній за сукупністю два правил, використовується E.CFVA. "Випадок" двох правил з використанням E.CFVA узагальнюється на будь-яку кількість правил.

Припустимо, що "а" і "в" - це два ЧУ, що сполучаються в один чинник на основі використання змінного середовища E.CFJO. Результуючий ЧУ приведений в табл. 1.

Таблиця 1.

Методи обчислення ЧУ на основі значення E.CFJO

Значення	Опис	Метод обчислення
M	Мінімальне значення	MIN(а, в)
P	Добуток	а*в/100
A	Середнє значення	(MIN(а, в)+(а*в/100))/2

З цих методів метод мінімального значення завжди в результаті даватиме найвищий ЧУ, а добуток - самий нижчий, середнє значення - це середнє між цими двома методами.

Якщо через E.CFJO з'єднуються більше двох ЧУ, то результати двох граничних методів обчислення (мінімум і добуток) не залежать від порядку обчислення цих чинників. Наприклад, об’єднання "с" результатом "а" і "в" дає той же результат, що і обчислення "в" з результатом "а" і "с".

Порядок обчислення може впливати на результат тільки у разі використання методів обчислення середнього значення і залишку, але ці результати завжди розташовуватимуться між граничними результатами (мінімум і добуток).

Припустимо "а" і "в" - це ЧУ, що сполучаються в один чинник на основі використання змінного середовища E.CFCO. Результуючий ЧУ приведений в табл. 2.

Таблиця 2

Методи обчислення ЧУ на основі E.CFCO

Значення	Опис	Метод обчислення
M	Максимальне значення	MAX(a, в)
P	Вірогідна сума	a + в – a*в/100
A	Середнє значення	(MAX(a, в)+(a+в-a*в/100))/2

З цих методів нижчий ЧУ завжди у методу максимуму, а вірогідна сума завжди має в результаті вищий ЧУ. Середнє значення розташовується посередині значень цих методів.

Якщо через E.CFCO з'єднуються більше двох ЧУ, то результати двох граничних методів обчислення (максимальне значення і вірогідна сума) не залежать від порядку обчислення. Наприклад, об'єднання "c" з результатом "a" і "в" дає в той же результат, що і обчислення "в" з результатом "a" і "c".

Порядок обчислення може впливати на результат тільки у разі використання методів середнього значення, але ці результати завжди розташовуватимуться між граничними результатами (максимальних значень і суми вірогідності).

Методи обчислення ЧУ для змінній E.CFVA

Під час консультації може використовуватися багато правил. Можливо, що декілька з них в результаті дадуть одне і те ж значення для певної змінної.

Наприклад, два зі всіх включених правил можуть мати LET URATE=0.06 CCF X як частина дій, де ЧУ = X тому, що URATE рівний 0.06, може бути для кожного правила різній. Більш того, кожне з правил може включатися з різним ступенем упевненості (тобто ЧУ їх посилок відрізняються один від одного).

"GURU" задає всі ці ЧУ для обчислення упевненості, при якій URATE рівний 0,06. Використовуваний метод обчислення визначається значенням CFVA, якщо тільки який-небудь інший метод обчислення не був специфікований для URATE в розділі змінних набору правив. У розділі змінних може використовуватися будь-якій з шістнадцяти типів обчислення, дозволений для E.CFVA.

Припустимо, що ai - це ЧУ, обчислений "GURU" для посилки правила i, а вi - це ЧУ, привласнений певному значенню змінної V при дії цього правила. Припустимо що aj - це ЧУ, який "GURU" обчислила для посилки правила j, а вj - це ЧУ, привласнений тому ж значенні V при дії правила j. "GURU" обчислює ЧУ для значення V в два етапи.

1. В межах правила "GURU" обчислює Ci як загальний ЧУ з ai і вi, Cj - як загальний ЧУ з aj і вj. Якщо припустити, що крім правил i і j зустрілися правила, в діях яких V привласнено теж саме значення, що і в діях правил i і j, то ЧУ кожного з цих правил обчислювався б тим же способом. При обчисленні загальної упевненості можуть послідовно використовуватися методи (M) МІНІМУМ, (P) ДОБУТОК або (A) СЕРЕДНЄ ЗНАЧЕННЯ.

2. За сукупністю правил "GURU" обчислює чинник упевненості для змінній V з Ci і Cj (і інших, якщо такі є) методом підтвердження. Даний спосіб обчислення використовує наступні методи: M (максимальне значення). P (сума вірогідності), A (середнє значення). Цей етап пропускається, якщо було включено хоч би одне правило, що дає в результаті певне значення V.

Вбудовані методи обчислення чинника упевненості для значення змінної під час консультації приведені в табл. 3

Таблиця 3

Вбудовані методи обчислення чинника упевненості

Значення	Опис	Обчислення в межах правила	Обчислення за правилом
PP	Сума ймовірностей добутку	Ci=ai*bi/100 Cj=aj*bj/100	Ci+Cj-Ci*Cj/100
PM	Сума ймовірностей мінімальних значень	Ci=MIN(ai,bi) Ci=MIN(aj,bj)	Ci+Cj-Ci*Cj/100
PA	Сума ймовірностей середніх значень	Ci=(MIN(ai,bi)+ +(ai*bi/100))/2 Cj=(MIN(aj,bj)+ +(aj*bj/100))/2	Ci+Cj-Ci*Cj/100
PB	Сума ймовірностей лишків	Ci=(ai * bi/100). (2-MAX(ai,bi)/100) Cj=(aj * bj/100). (2-MAX(aj,bj)/100)	Ci+Cj-Ci*Cj/100
MP	Максимальне число добутків	Ci=ai*bi/100 Cj=aj*bj/100	MAX(Ci,Cj)
MM	Максимальне число мінімальних значень	Ci=MIN(ai,bi) Ci=MIN(aj,bj)	MAX(Ci,Cj)
MA	Максимальне число мінімальних значень	Ci=(MIN(ai,bi)+ +(ai*bi/100))/2 Cj=(MIN(aj,bj)+ +(aj*bj/100))/2	MAX(Ci,Cj)
MB	Максимальне число лишків	Ci=(ai * bi/100). (2-MAX(ai,bi)/100) Cj=(aj * bj/100). (2-MAX(aj,bj)/100)	MAX(Ci,Cj)
AP	Середнє число добутків	Ci=ai*bi/100 Cj=aj*bj/100	((Ci+Cj-Ci*Cj/100)+ +MAX(Ci,Cj))/2

AM	Середнє число мінімальних значень	Ci=MIN(ai,bi) Ci=MIN(aj,bj)	((Ci+Cj-Ci*Cj/100)+ +MAX(Ci,Cj))/2
AA	Середнє число середніх значень	Ci=(MIN(ai,bi)+ +(ai*bi/100))/2 Cj=(MIN(aj,bj)+ +(aj*bj/100))/2	((Ci+Cj-Ci*Cj/100)+ +MAX(Ci,Cj))/2
AB	Середнє число лишків	Ci=(ai * bi/100). (2-MAX(ai,bi)/100) Cj=(aj * bj/100). (2-MAX(aj,bj)/100)	((Ci+Cj-Ci*Cj/100)+ +MAX(Ci,Cj))/2
BP	Лишок добутку	Ci=ai*bi/100 Cj=aj*bj/100	MAX(Ci,Cj)+MIN (Ci,Cj)*(1-Ci/100)(1- Cj/100)
BM	Лишок мінімальних значень	Ci=MIN(ai,bi) Ci=MIN(aj,bj)	MAX(Ci,Cj)+MIN (Ci,Cj)*(1-Ci/100)(1- Cj/100)
BA	Лишок середніх значень	Ci=(MIN(ai,bi)+ +(ai*bi/100))/2 Cj=(MIN(aj,bj)+ +(aj*bj/100))/2	MAX(Ci,Cj)+MIN (Ci,Cj)*(1-Ci/100)(1- Cj/100)
BB	Лишок лишків	Ci=(ai * bi/100). (2-MAX(ai,bi)/100) Cj=(aj * bj/100). (2-MAX(aj,bj)/100)	MAX(Ci,Cj)+MIN (Ci,Cj)*(1-Ci/100)(1- Cj/100)

Значення ЧУ для виразів, що містять змінні

Нехай дані команди:

FIRSTHAME = "ГЕОРГІЙ" cf 40
LASTNAME = "ГЕОРГИЕВ" cf 60
NAME = FIRSTNAME + LASTNAME

"GURU" повинна скористатися яким-небудь методом, щоб обчислити загальний ЧУ для NAME. Метод обчислення ЧУ виразів залежить від E.CFJO. За умовчанням E.CFJO = M (мінімум). В цьому випадку NAME має ЧУ, рівний 40. Інший приклад:

FIRSTNAME = "ГЕОРГІЙ" cf 40
LASTHAME = "ГЕОРГИЕВ" cf 60
NAME = FIRSTNAME + LASTNAME cf 80.

Цей приклад аналогічний попередньому, за винятком додаткового третього ЧУ, який ускладнює процес обчислення загального ЧУ. Спочатку за допомогою E.CFJO об'єднуються ЧУ FIRSTNAME і LASTNAME. Видається проміжний ЧУ, рівний 40. ЧУ, рівний 40, об'єднується з ЧУ змінної NAME, рівний 80. Знову береться мінімум і вибирається 40.

Змінні набору (нечіткі змінні)

Змінна; яка одночасно може мати декілька значень, кожне з яких характеризується своїм чинником упевненості, називається змінною набору. За допомогою цих змінних можна одержувати якнайкращу консультацію: ви можете використовувати їх для того, щоб знайти якнайкращий варіант або найбільш вірогідне значення для критеріїв ухвалення рішення.

Наприклад, якщо А має 2 значення, В - 3 значення, то X = A + B може мати 3 значення. Дублюючі результуючі значення об'єднуються. Якщо у вас є команда "variable = 5", то вона може означати або те, що зараз змінна має значення 5, або те, що окрім всіх попередніх значень, що приймаються цією змінною, вона має зараз значення 5 з ЧУ 100.

Щоб розрізняти ці випадки у "GURU" існує синтаксична структура {…}, яка використовується для того, щоб вказати набір значень, що належать однією змінною. X = { } - означає, що X немає відомих значень. X = {1,2,3} - змінна X є змінною набору, що має три значення 1, 2, 3.

Змінні набору можуть містити значення, що відносяться до різних типів. Наприклад {1, "яблуко", 2.1}. Кожен елемент змінної набору може мати свій ЧУ, наприклад: {1 cf 30, 2 cf' 40, 3 cf 100}.

Всі елементи набору переносяться в привласнюване значення, наприклад: X = {1, 2 cf 50, 3}, У = X, тоді У = {1, 2 cf 50, 3}.

Задавання можна виконувати і за умови невизначеності, наприклад: X={1, 2, 3}, Y = X cf 50; тоді Y = {1 cf 50, 2 cf 50, 3 cf 50}.

Використовуючи E.CFCC, ЧУ привласнених значень можна об'єднати.

Припустимо, що E.CFCO - "P". В цьому випадку такі записи: X = {1 cf 50. 2 cf 60}, У = X cf 50 можна замінити на: У = {1 cf 25, 2 cf 30}.

У "GURU" існують оператори "+ =" і "- =', які додають елементи в набір і видаляє елементи з набору. Наприклад, запис: X = {1, 2, 3 } і X + = 4 означає, що X = {1, 2, 3, 4}.

Використовуючи оператор "+ =" можна об'єднати, наприклад, X={1,2,3}, Y = { 4,5 }, X += Y, тоді X = {1,2,3,4,5}. Ще приклад: X = {1,2,3}, X -= {2,3}, тоді X = {1}.

Функції пошуку ЧУ

CFV (змінна, значення) - видає ЧУ даного значення змінної.

CFN (змінна, номер) - видає ЧУ значення змінної, вказаного номером. Порядок наступний: 1 - найвищий ЧУ, 2 - другий за величиною і т.д.

VALN (змінна, номер) - видає значення змінної.

NUMVAL (змінна) - видає кількість значень змінної.

HIVAL (HIVAL змінна) - видає значення змінної з найвищим ЧУ.

HICF (змінна) - видає ЧУ значення HIVAL.

LOVAL (змінна) - видає значення змінної з найменшим ЧУ.

LOCF (змінна) - видає значення змінної LOVAL.

TAB (змінна) - видає внутрішній номер змінної, який може використовуватися командою HOW.

KNOWN (змінна) - видає true, якщо у змінною є значення, ЧУ яких

перевищує мінімальний рівень достовірності, що встановлюється за допомогою E.UNKN. Наприклад, геолог упевнений в тому, що скельний пласт можна швидше виявити на глибині 20 футів, чим на глибині 40 футів, а знайти його на глибині 60 футів майже неможливо. Тоді BERDOCK = {60 cf 5, 40 cf 62, 20 cf 89}.

Інший приклад. Припустимо, колір (color) має значення: червоний cf 40, жовтий cf 60, блакитний cf 50, а будинок (house) має значення червоний cf 70.

Таблиця 3.

Приклади обчислення ЧУ для різних посилок

Посилка	ЧУ посилки
Якщо колір = червоний	40
Якщо колір = червоний або колір = блакитний	40 “або” 50
Якщо колір не червоний	60 “або” 50
Якщо будинок = червоний	70
Якщо будинок не червоний	FALSE

КОНТРОЛЬНІ ПИТАННЯ

1. Які змінні управляють обчисленням ЧУ і як це вони роблять?
2. Які відмінності в застосуванні E.CFJO, E.CFCO і E.CFVA?
3. Що роблять операторів "+=" і "- ="?
4. Як відбувається обчислення ЧУ в змінних набору?
5. Які функції визначають значення ЧУ змінній?
6. Навіщо потрібна змінна E.UNKN?

ПРИКЛАД РОЗВ’ЯЗАННЯ ТИПОВОГО ЗАВДАННЯ

Припустимо, що ми проводимо технічний огляд автомобіля і хочемо зробити висновок, поїде він чи ні. Ми перевіряємо акумулятор, стартер і ланцюг запалення і на основі результатів огляду ухвалюємо рішення. Ось які правила в цій експертній системі.

Правило 1. Якщо справні стартер, акумулятор і ланцюг запалення, то двигун заведеться з ЧУ, рівним мінімуму ЧУ акумулятора, стартера і ланцюга запалення.

Правило 2. Якщо ЧУ того, що двигун заведеться, більше 30, то система запалення справна.

Правило 3. Якщо ЧУ того, що двигун заведеться, більше 30, але менше 80, то необхідний дрібний ремонт.

Правило 4. Якщо ЧУ того, що двигун заведеться, менше 30, то необхідний капітальний ремонт.

От як виглядає ця ЕС на мові "GURU":

GOAL: RESULT

RULE: R1
IF: AKKUM = "ЗАРЯДЖЕНИЙ" AND START = "СПРАВНИЙ" AND CIR = "СПРАВНИЙ"
THEN: DVIG = "ЗАВЕДЕТЬСЯ"
REASON: якщо стартер, акумулятор і ланцюг запалення у справному стані, то двигун заведеться.

RULE: R2
IF: HICF(DVIG) > 80
THEN: RESULT = "СПРАВНИЙ"
RSASGH: якщо ЧУ того, що двигун заведеться, більше 60, то автомобіль – справний.

RULE: R3
IF: HICF(DVIG) >= 30 AND HICF(DVIG) <= 80
THEN: RESULT = "ДРІБНИЙ РЕМОНТ"
REASON: якщо ЧУ того. що двигун заведеться більше 30 і менше 80, то необхідний дрібний ремонт.

RULE: R4
IF: HICF(DYIG) < 30
THEN: RESULT = "капітальний ремонт"
REASOH: якщо ЧУ того. що двигун заведеться менше 30, то необхідний капітальний ремонт

INITIAL:
E.RIGR = "A"
E.WHN = "F"
E.TRYP = "P"
E.LSTR = 80
E.DECI = 0
E.CFVA = "MM"
E.CFJO = "M"
E.CFCO = "M"
START = UNKNOWN
CIR = UNKNOWN
AKKUM = UNKNOWN
DVIG = UNKNOWN

```
RESULT = UNKNOWN
PROM = UNKNOWN
CLEAR «*** ТЕХОГЛЯД АВТОМОБІЛЯ ***»
OUTPUT
OUTPUT "відповідайте на всі питання і одержите консультацію"
OUTPUT "про систему запалення"
OUTPUT

DO:
CLEAR
OUTPUT "КОНСУЛЬТАЦІЯ ПРО СИСТЕМУ ЗАПАЛЕННЯ", RESULT

VAR: AKKUM
FIND: PROM=0
INPUT PROM NUM WITH “на скільки ви упевнені, що акумулятор заряджений?”
AKKUM = “ЗАРЯДЖЕНИЙ” CF PROM
LABEL визначає чи заряджений акумулятор

VAR: CIR
INPUT PROM NUM WITH "наскільки ви упевнені, що схема запалення справна?"
CIR = “СПРАВНА” CF PROM
LABEL: визначає, чи справна схема запалення

VAR: START
FIND: PROM = 0
INPUT PROM NUM WITH "наскільки ви упевнені, що справний стартер?"
START = "СПРАВНИЙ"
LABEL: визначає, чи справний стартер

VAR: DVIG
LABEL: визначає, чи заведеться двигун

VAR: RESULT
LABEL: результат технічного огляду

VAR: PROM
LABEL: проміжна змінна для введення ЧУ

END:
```

Розглянемо детальніше цю ЕС. Метою є змінна RESULT. Правило R1 визначає технічний стан приладів. Правила R2, R3, R4 перевіряють граничні

значення ЧУ. Функція HICF() визначає максимальне значення ЧУ.

Тепер розглянемо змінні, які використовуються в ЕС.

Змінна PROM - це допоміжна змінна, яка служить для введення значень ЧУ. Змінні AKKUM, START і CIR визначаються однаковим чином.

Оскільки ці змінні повинні вводитися в ході діалогу, то вони мають пункт FIND. Тут спочатку змінною PROM привласнюється нуль, потім в змінну PROM вводиться значення ЧУ відповідній змінній, потім відповідна змінна спеціалізується з введеним ЧУ. Дані змінні повинні запрошуватися при перевірці першого правила.

При ініціалізації ЕС визначаються системні змінні.

E.RIGR = "A" - використання всіх необхідних правил.

E.WHN - "F" - для пошуку значення невідомої змінної. Пошук здійснюється ще до спроби вивести значення невідомих змінних. При цьому запрошується FIND.

E.TRYP - "P" - терпляче намагатися визначити значення всіх невідомих змінних, а потім оцінити посилку.

E.LSTR - 80 - довжина символьного рядка при висновку на екран.

E.DECI = 0 - кількість десяткових знаків.

E.CFVA = "MM", E.CFJO = "M", E.CFCO = "M" - визначають алгебру ЧУ. Далі йде ініціалізація змінних і команди для визначення роботи системи.

ПІДГОТОВКА ДО ВИКОНАННЯ ЗАВДАННЯ

1. Дайте відповіді на контрольні питання.
2. Ознайомтеся з прикладом розв’язання типового завдання.

ЗАВДАННЯ ДЛЯ САМОКОНТРОЛЮ

Додайте до ЕС, розробленій на практичному занятті №14, зміни, в яких застосовуються ЧУ. Для демонстрації варіанту таких змін в методичних вказівках до індивідуальної і самостійної роботи приведений приклад ЕС з використанням спеціальних оголошуваних змінних середовища і ЧУ.

ПОРЯДОК ВИКОНАННЯ ЗАВДАННЯ

1. Покажіть викладачу текст змін в ЕС, написаної і відладженої при виконанні практичної роботи № 14.
2. Запустите систему "GURU". Створіть базу знань на основі свого варіанту ЕС.
3. Перевірте за допомогою HOW і WHY роботу створеної ЕС. Результати покажіть викладачу.

ЗВІТ ПРО ВИКОНАНЕ ЗАВДАННЯ ПОВИНЕН МІСТИТИ:

1) відповіді на контрольні питання;
2) варіант ЕС на природній мові;
3) варіант ЕС на мові "GURU";
4) приклад відладки ЕС за допомогою HOW і WHY;
5) висновок.

ТЕМА 17. Інструментальні комплекси для побудови ЕС

Інструментальні комплекси для побудови статичних ЕС і ЕС реального часу. Інструментальні оболонки експертних систем.

ПРАКТИЧНЕ ЗАНЯТТЯ №16. МОВА КОМАНД ОБОЛОНКИ ЕС "GURU".

Мета: знайомство з командним режимом "GURU" і створення простих програм з використанням команд оболонки "GURU.

План заняття:

1. Загальна характеристика командної мови "GURU" (умовні переходи, функції, оператори, змінні тощо).
2. Відпрацювання навиків використання команд BOILD, COMPILE, CONSULT, RUN, DIR, LET, OUTPUT, INPUT, PERFORM, RETURN, WAIT, WHILE-DO, TEST, IF-THEH-ELSE, CONTINUE, BREAK, CLEAR, BYE.
3. Розбір прикладу розв’язання типового завдання.
4. Самостійне написання програми з використанням команд оболонки "GURU.

Теоретичні відомості, що необхідні для виконання даної роботи

Командний режим "GURU". Основні команди.

Інтегрована оболонка для створення експертних систем "GURU" має розвинену командну мову. Сюди входять багато конструкцій, які є і у МВР. Це умовні переходи, функції, різні види операторів, різні типи змінних і т.п.

Звичайно, спеціалізація "GURU" на створення ЕС наклала відбиток і на командну мову. Тому тут немає багатьох можливостей МВР, зате є особливості, необхідні при створенні ЕС. Покажемо, як можна створювати ЕС, користуючись командною мовою.

***Команда* BUILD**

Щоб створити набір правил з командного рядка, дайте команду BUILD <ім'я набору правил>. Після цього на екрані з'являється основне меню адміністратора набору правил системи "GURU".

Приклад: BUILD MYEXPERT За допомогою цієї команди створюється набір правил MYEXPERT і відкривається початковий файл набору правил, названий MYEXPERT.RSS.

***Команда* COMPILE**

Команда COMPILE <ім'я набору правил> створює на основі тексту набору правил експертну систему, яку можна виконати.

Приклад: COMPILE MYEXPERT В цьому випадку компілюється початковий файл набору правил MYEXPERT.RSS, який створює файл експертної системи MYEXPERT.RSC. Всі помилки і попередження з'являються на екрані і заносяться у файл MYEXPERT.WRN.

***Команда* CONSULT**

Команда CONSULT <ім'я експертної системи> застосовується для консультації з експертною системою. Більш розширене використання

команди CONSULT:

CONSULT [набір правил] [метод [об'єкт]] [метод [об'єкт]],

визначає для машини логічних висновків метод обробки; [об'єкт] - змінна мети, правило або список правил.

Можливі наступні чотири методи:

TO SEEK - виконує обробку, використовуючи для визначення мети метод зворотного зчеплення;

TO TEST - виконує обробку, використовуючи для визначення мети метод прямого зчеплення;

TO FIRE - виконує обробку, використовуючи для запуску даного правила метод зворотного зчеплення;

TO EXECUTE - виконує обробку списку даних правил, використовуючи метод прямого зчеплення.

Різні методи допускають використання різних об'єктів. Для методів "SEEK" і "TEST" об'єктом є ім'я змінної мети. Якщо ця змінна не вказана, тоді використовується змінна мети, що міститься в наборі правил; для методу "FIRE" об'єктом є ім'я окремого правила "GURU", яке "GURU" спробує запустити. У разі потреби використовується аргументування за методом зворотного зчеплення. За умовчанням, як об'єкт береться перше правило в наборі. Для методу "EXECUTE" об'єкт - це список правил. Ці правила запускаються по порядку із застосуванням методу прямого зчеплення, якщо не враховувати того, що виконується тільки один прогін.

Приклади використання команди CONSULT:

CONSULT MYEXPERT TO TEST MYGOAL - для знаходження мети MYGOAL використовується метод прямого зчеплення;

CONSULT MYEXPERT TO FIRE 1 - робиться спроба запустити правило RULE1.

***Команда* RUN**

Виконує зовнішні програми (*.EXE і *.COM), що працюють під управлінням MS DOS. Система "GURU" відновлює обробку після того, як зовнішня програма виконана. Приклади:

RUN COPY A: MYFILE.TXT C: - копіює файл MYFILE.TXT з диска A на диск C;

RUN PROG.EXE - запускає програму PROG.

***Команда* DIR**

Огляд існуючих файлів в директорії. Приклад: за допомогою команди DIR C:\MYDIR\ здійснюється огляд файлів в директорії MYDIR на диску C.

***Команда* LET**

Надає значення змінним. Цю команду треба ввести у форматі: LET <змінна> = < вираз >. Тут:

< змінна > - робоча змінна, змінна поля або змінна середовища;

< вираз > - константа, змінна і оператори, які виражають символьне, числове або логічне значення.

Примітка: слово LET необов'язково.

Приклади:
LET MYPER1 = "ABCD"
(аналогічно: MYPER1 = "ABCD")
Привласнює змінною HYPER1 значення "ABCD".
MUPER2 = MUPER1
Привласнює змінною MUPER2 значення змінної MUPER1.

***Команда* OUTPUT**

Призначена для виведення повідомлення на термінал. Має формат:

OUTPUT < вираз > [тип] [using < шаблон >]

Тут: < вираз > - константа, вираз, змінна; [тип] - тип виразу, що виводиться (необов'язковий). Може бути "mm", "str", "log".

using < шаблон > - шаблон виразу, що виводиться.

Приклади: OUTPUT MYPER1 - виводить змінну MYPER1.

OUTPUT MYPER1 using "aaa" - виводить три перші буквені символи змінної MYPER1.

***Команда* INPUT**

Служить для введення. Має формат:

INPUT < змінна > [тип] [using < шаблон >] [with < підказка >].

Тут: < змінна > - див. коментар до команди OUTPUT;

тип - див. коментар до команди OUTPUT; using < шаблон > - див. коментар до команди OUTPUT; with < підказка > - виводить текст підказки.

Приклади:
INPUT MYPER1 - вводиться з екрану значення в MYPER1.
INPUT MYPER1 using "aaa" - вводяться тільки три букви.

INPUT MYPER1 with "Введіть рядок" - виводиться підказка "Введіть рядок" і запит на введення в змінну MYPER1 якого-небудь значення.

***Команда* PERFORM**

Цією командою запускається програма "GURU". Команда має формат: PERFORM "ім'я файлу". Тут: "ім'я файлу" - файл з командами "GURU", що має розширення *.IPF. Приклад: PERFORM MYPROGRAM - запускає програму MYPROGRAM.

***Команда* RETURN**

Закінчує виконання процедури і забезпечує повернення в ту точку, з якої викликалася програма, або в командний режим. Має формат: RETURN.

***Команда* WAIT**

Зупиняє виконання програми до тих пір, поки користувач не натисне яку-небудь клавішу. Має формат: WAIT.

***Команда* WHILE-DO**

Створює цикл, що складається з команд "GURU". Цикл обробляється до тих пір, поки не будуть задоволені задані умови. Має формат:

WHILE <умови> DO <твердження> ENDWHILE.

Тут: <умови> - критерій, специфікований в термінах логічних виразів; < твердження > - послідовність команд "GURU".

***Команда* TEST**

Перевіряє значення специфікованого виразу для певного варіанту. Має формат:

TEST < вираз >
CASE < вираз - 1: > < твердження - 1 >
CASE < вираз - 2: > < твердження - 2 >
:
:
OTHERWISE: < твердження >
ENDTEST.

Тут: <вираз> - одна або декілька констант і/або змінних, об'єднаних операторами, які обчислюють символьне, числове або логічне значення (TRUE або FALSE); < твердження > - послідовність команд "GURU"; OTHERWISE необов'язково.

***Команда* IF-THEN-ELSE**

Реалізує перший набір команд, якщо задовольняються вказані умови, або інший набір команд, якщо умови не задовольняються. Має формат:

IF < умови > THEN < твердження > ELSE < твердження > ENDIF.

Тут: < умови > - критерії, вказані на основі логічного виразу;
< твердження > - набір команд "GURU".

УВАГА! IF-THEN-ELSE не має нічого спільного з правилами (RULE) "GURU".

***Команда* CONTINUE**

Зупиняє обробку части DO команди WHILE-DO і повторно оцінює умови WHILE.

***Команда* BREAK**

Зупиняє обробку DO-WHILE або забезпечує вихід з TEST.

***Команда* CLEAR**

Очищає екран.

***Команда* BYE**

Забезпечує вихід з командного режиму.

Команди текстового редактора.

TEXT <ім'я файлу> - запустити текстового редактора "GURU";

READ <ім'я файлу> - припинити обробку поточного тексту і почати обробку іншого текстового файлу без зупинки і повторного виклику текстового редактора;

TOP - пересунути курсор у верхню частину поточного тексту;

BOTTOM - пересунути курсор в нижню частину поточного тексту;

GOTO <позиція> - пересунути курсор в певний рядок тексту;

SEARCH - пошук певної групи символів в тексті;

INSERT <ім'я файлу> - вставити вміст файлу в поточний текст;

PRINT - друкувати текст;
QUIT - вийти з текстового редактора без збереження змін;
STOP - вийти з текстового редактора із збереженням;
<CTRL-l> - допомога.

КОНТРОЛЬНІ ПИТАННЯ

1. Назвіть переваги і недоліки командного режиму.
2. Які команди "GURU" використовуються для логічних переходів за умовами?
3. Якими способами можна консультуватися з набором правил? Приведіть приклади.
4. Яким чином можна запустити зовнішні команди MSDOS? Приведіть приклади.
5. Синтаксис і семантика команди CONSULT.
6. Команди текстового редактора.
7. Команди утворення циклів.
8. Призначення команди TEST.
9. Команди введення-виведення.
10. Команди виконання програм.

ПРИКЛАД РОЗВ'ЯЗАННЯ ТИПОВОГО ЗАВДАННЯ

Описана нижче програма виводить на екран меню, з якого можна виконати наступні дії:

1) запустити редактора набору правил;
2) запустити консультацію своєї ЕС;
3) вийти з програми в командний режим.

Текст програми:

```
/* Файл MYPROG.IPF */
e.lstr = 80 /* довжина символьного поля */
while true do /* задає нескінченний цикл */
clear /* очищає екран */
choise =" "
output "Головне меню"
output "--------------------"
output "Побудова і редагування правил"
output "Консультація із створеним набором правил"
output "Вихід в командний режим"
output "--------------------"
input choise using "c" with "Ваш вибір?"
test choise /* перевірка вибору і виконання */
/* одного з наступних розділів */
      case "1":
            build MYEXPERT /* побудувати ЕС MYEXPERT */
```

```
            break
        case "2":
            consult MYEXPERT to test /* консультація */
            /* з MYEXPERT */
            break
        case "3":
            return
            break
        otherwise:
            output "Неправильно введений символ"
            output "Натисніть будь-яку клавішу"
            wait
            break
    endtest
    endwhile
```

Цикл while-do необхідний для того, щоб після вибору якого-небудь пункту меню, окрім "Вихід", ми поверталися знову в головне меню. Для запуску програми запустите з командного рядка: PEPFORM MYEXPERT

ПІДГОТОВКА ДО ВИКОНАННЯ ЗАВДАННЯ

1. Дайте відповіді на контрольні питання.
2. Ознайомтеся з прикладом розв’язання типового завдання.
3. Відповідно до заданого варіанту, напишіть програму на мові "GURU".

ПОРЯДОК ВИКОНАННЯ ЗАВДАННЯ

1. Пред'явите викладачу написану при виконанні домашнього завдання програму з використанням командної мови "GURU".
2. Запустите систему "GURU".
3. Виберіть в меню режим "Зміна середовища".
4. Виберіть в меню підрежим "Командний режим". Після цього з'являється підказка для введення команд.
5. Введіть команду BUILD <ім'я свого набору правил>. Далі створіть або відредагуйте набір правил (аналогічно тому, як ви робили це в роботі 13).
6. За допомогою CONSULT проконсультуйтеся зі своїм набором правил.
7. Введіть команду TEXT, яка викликає текстовий редактор "GURU".
8. Введіть свою підготовлену програму. При цьому скористуйтеся командами текстового редактора.
9. Перевірте правильність роботи програми за допомогою PERFORM <ім'я програми>.
10. Результати роботи програми покажіть викладачу.

ЗВІТ ПРО ВИКОНАНЕ ЗАВДАННЯ ПОВИНЕН МІСТИТИ:

1) відповіді на контрольні питання;
2) програму на мові "GURU" згідно варіанту завдання;
3) результати роботи програми;
4) підсумок, в якому необхідно зробити висновки щодо командного режиму "GURU", про його переваги і недоліки, порівняти з мовами високого рівня і з діалоговим режимом роботи.

ЗАВДАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Написати програму, яка виводить головне меню на екран, з якого можна викликати наступні дії: 1) побудова і редагування свого набору правил; 2) консультація з одним правилом з набору правил; 3) вихід в командний режим.

2. Написати програму, яка виводить головне меню на екран, з якого можна викликати наступні дії: 1) побудова і редагування свого набору правил; 2) консультація з набором правил методом прямого аргументування; 3) вихід в командний режим.

3. Написати програму, яка виводить головне меню на екран, з якого можна викликати наступні дії: 1) побудова і редагування свого набору правил; 2) консультація з набором правил, при якій визначається не змінна мети, а яка-небудь проміжна змінна мети; 3) вихід в командний режим.

4. Написати програму, яка виводить головне меню на екран, з якого можна викликати наступні дії: 1) побудова і редагування свого набору правил; 2) виклик текстового редактора; 3) вихід в командний режим.

5. Написати програму, яка виводить головне меню на екран, з якого можна викликати наступні дії: 1) побудова і редагування свого набору правил; 2) консультація з набором правил методом зворотнього аргументування; 3) вихід в командний режим.

6. Написати програму, яка виводить головне меню на екран, з якого можна викликати наступні дії: 1) побудова і редагування свого набору правил; 2) виклик певної зовнішньої програми; 3) вихід в командний режим.

7. Написати програму, яка виводить головне меню на екран, з якого можна викликати наступні дії: 1) побудова і редагування свого набору правил; 2) компілювання та виконання програми; 3) вихід в командний режим.

8. Написати програму, яка виводить головне меню на екран, з якого можна викликати наступні дії: 1) побудова і редагування свого набору правил; 2) виклик іншої експертної системи; 3) вихід в командний режим.

9. Написати програму, яка виводить головне меню на екран, з якого можна викликати наступні дії: 1) побудова і редагування свого набору правил; 2) виклик команда DIR; 3) вихід в командний режим.

ТЕМА 18. Сучасний стан та перспективи розвитку ЕС

Експертні системи реального часу - основний напрям розвитку систем штучного інтелекту. Характеристика оболонок експертних систем реального часу як самодостатніх середовищ для розробки, впровадження і супроводу додатків в широкому діапазоні галузей. Універсальні технології побудови сучасних ЕС (стандарти відкритих систем, архітектура клієнт/сервер, об'єктно-орієнтоване програмування, використання ОС, що забезпечують паралельне виконання в реальному часі багатьох незалежних процесів). Спеціалізовані методи ЕС (міркування, засновані на правилах, міркування, засновані на динамічних моделях або імітаційне моделювання, процедурні міркування, активна об'єктна графіка, структурована природна мова для представлення бази знань).

ПРАКТИЧНЕ ЗАНЯТТЯ №17. ЕЛЕКТРОННІ ТАБЛИЦІ ОБОЛОНКИ ЕС "GURU".

Мета: ознайомитися з основними можливостями та сформувати навички використання команд електронних таблиць оболонки "GURU".

План заняття:

1. Електронні таблиці "GURU".
2. Формування навичок використання команд EB\DUMP, EB\DISPLAY, EB\STOP, EB\LOAD, EB\SAVE, EB\COPY, EB\LET, EB\COMPUTE, EB\WIDTH, EB\USING, EB\UNDEFINE, EB\EDIT.
3. Розбір прикладу розв’язання типового завдання.
4. Створення власної ЕС.

Теоретичні відомості, що необхідні для виконання даної роботи

Часто при виконанні розрахунків зручними є початкові дані і результати розрахунків у вигляді таблиці. Для автоматизації табличних розрахунків використовуються спеціальні види програмного забезпечення, названі електронними відомостями (ЕВ).

Електронна відомість "GURU" - це робоча таблиця, яка спрощує процес роботи з числами. ЕВ - це матриця величиною 255 на 255 елементів, в якої цифри ідентифікують рядки, а букви - стовпці. Перетин рядка і стовпця називається осередком.

Кожен осередок позначається знаком #, за яким слідує вказівка місцеположення стовпця або рядка (наприклад, #A1 визначає осередок у верхньому лівому кутку ЕВ).

При вході в режим обробки ЕВ можна давати всі команди, визначені "GURU". При цьому перед кожною командою необхідно давати символ "\".

Наприклад: \CLEAR \PERFORM MY_PROGRAM \OUTPUT "Це команда OUTPUT".

У "GURU" є додаткові команди, визначені тільки для ЕВ.

Режим обробки ЕВ

Для того, щоб увійти до режиму обробки ЕВ, необхідно ввести в командному режимі команду CALC. Синтаксис цієї команди:

CALC < розмірність >,

де < розмірність > (ЕВ) - два вирази, відокремлені комами, вказуючи число рядків і число стовпців.

Якщо розмірність відсутня, то за умовчанням визначається ЕВ розміром 30 рядків на 30 стовпців. Наприклад, CALC 10,20 - визначає ЕВ розміром 10 рядків і 20 стовпців.

Після введення команди CALC на екран виводяться межі і осередки ЕВ, область стану і рядок введення. Область стану складається з:

- допоміжного рядка (перший рядок); використовується для більш довершеної обробки ЕВ;
- рядки введення (другий рядок); використовується для вказівки подальших дій і введення команд;
- рядки висновку (третій рядок); виводить визначення (якщо воно існує) поточного осередку, а також виводить повідомлення про помилки.

Якщо в рядку введення ввести \< команда >, то виконується команда обробки ЕВ. Якщо ж коса межа (\) відсутня, то "GURU" вважає, що вводиться визначення осередку.

Визначення осередку означає, що осередку можна привласнити значення константи або яку-небудь функцію. Наприклад, осередок може мати визначення sin(#A2) - це означає, що осередок має значення синуса вмісту осередку #A2. Коли задається команда розрахунку ЕВ, то відразу виконується ця функція.

Якщо в командному рядку вводиться \<команда>, то "GURU" виводить повідомлення: "*ЗАНЯТО*". Коли "GURU" закінчує обробку команди, то це повідомлення зникає.

Клавішами управління курсором можна переміщатися від одного осередку до іншого. Поточний осередок буде виведений на екрані.

Команди ЕВ

***Команда* ЕВ \EDIT**

\EDIT - вивести визначення поточного осередку, який надалі може бути відредагований або змінений за допомогою клавіш управління курсором. Встановите курсор на осередку, який необхідно редагувати. Після команди \EDIT можна змінити визначення цього осередку.

***Команда* ЕВ \UNDEFINE**

\UNDEFINE - видалити існуюче визначення осередку, що включає значення, тип представлення і шаблон осередку. Ввести \UNDEFINE <осередок> або \UNDEFINE <блок осередків>, де <осередок> - ім'я осередку, <блок осередків> - блок осередків. Специфікуються двома осередками, які указують верхній лівий і нижній правий кути блоку осередків. Якщо осередок (або блок осередків) опущені, то віддаляється визначення

поточного осередку.

Приклади:

\UNDEFINE #A4 - видалити визначення осередку #A4,

\UNDEFINE - видалити визначення поточного осередку,

\UNDEFINE #D8 #L25 - видалити визначення блоку осередків з #D8 по #L25.

***Команда* EB \USING**

Задає шаблон осередку або блоку осередків.

\USING "шаблон" < осередок >

\USING "шаблон" < блок осередків >

"шаблон" - специфікує редагування, яке повинно здійснювати "GURU";

<осередок> або <блок осередків>- див. опис команди EB "UNDEFINE".

Приклади:

\USING "(ddd)ddd-ddd" #C5

\USING "dd-dd" #A4 #G8

Всі дані, які будуть вводиться в цей осередок, будуть перетворені за шаблоном.

***Команда* EB \WIDTH**

Задає ширину стовпців. За умовчанням ширина стовпців рівна 9.

\WIDTH <ширина стовпця>.

Мінімальна ширина стовпця - 5, максимальна - 255.

Приклади:

WIDTH #E20 - встановити ширину стовпця #E рівною 20.

***Команда* EB \COMPUTE**

Обчислює всі значення осередків EB, засновані на поточних визначеннях.

\COMPUTE або \COMPUTE <блок осередків>

Команда, введена без параметрів, обчислює значення всіх осередків EB. Приклади: \COMPUTE #D12 #M37

***Команда* EB \LET**

Привласнює змінною значення.

\LET< змінна >=< вираз >USING"шаблон".

Слово "LET" можна опустити.

Приклади:

\LET #C5=9

\LET #A16=#D18+#D20

\#F12=SNUM USING "DD-DD-DDDD".

***Команда* EB \COPY**

Копіює визначення одного осередку в іншій.

\COPY <початковий осередок>, < цільовий осередок >.

"GURU" висвічує на екрані: "Вирівнювати (Y/N)?" Якщо Y, то "GURU" встановлює вираз щодо цільового осередку. Якщо N, то змінна вказаного осередку залишається незмінного у визначенні цільового осередку.

Приклад: \COPY #A20, #C30 - копіює визначення осередку #A20 в осередок #C30.

***Команда* EB \SAVE**

Зберегти EB у файлі. \SAVE TO "файл"

Приклад: \SAVE TO TABLE - зберегти EB у файлі TABLE.

***Команда* EB \LOAD**

Завантажити EB з файлу. \SAVE FROM "файл".

Приклад: \SAVE FROM TABLE - завантажити EB з файлу TABLE.

***Команда* EB \STOP**

Закриває EB і повертається в командний режим "GURU".

Приклад: \STOP.

***Команда* EB \DISPLAY**

Виводить на друк необхідні ділянки або всі значення осередків EB.

\DISPLAY <блок осередків>

***Команда* EB \DUMP**

Друкує всю інформацію щодо заданих осередків, яка включає: ім'я осередку, поточне значення, визначення (якщо існує) і шаблон (якщо існує).

\DUMP < блок осередків >.

Використання EB в програмі

Осередки EB можуть виступати як змінні в програмі. Ім'я осередку повинне починатися з символу "#". Осередку EB можна привласнити яке-небудь значення. Приклад: #C2=18.

КОНТРОЛЬНІ ПИТАННЯ

1. Яким чином вводяться команди "GURU" в режиму обробки EB?
2. Що таке визначення і значення осередку EB?
3. Як можна дістати з "GURU" визначення і значення осередків EB?
4. Якими командами завантажується EB з файлу і записується у файл?
5. Якими командами обробляються осередки EB?
6. Як отримати інформацію щодо заданих осередків?
7. Як задати ширину стовпців EB?
8. Як увійти до режиму обробки EB?

ПРИКЛАД РОЗВ'ЯЗАННЯ ТИПОВОГО ЗАВДАННЯ

Створити EB наступної структури для розрахунку заробітної платні:

	A	B	C	D	E
1	Прізвище	Стать	Оклад	Відпрацьовано днів	Нараховано
2	_____	____	_____	_____	_____
3	Іванов	чоловік	(000)	( )	( )
4	Смирнова	жінка	(000)	( )	( )
5	Ковальов	чоловік	(000)	( )	( )
6	_____	____	_____	_____	_____

Розв'язання. Опишемо послідовність дій для створення даної ЕВ.

1. Командою CALC увійдемо до режиму обробки ЕВ.

CALC 6, 5 - створюється ЕВ розміром 6 рядків і 5 стовпців.

2. Командою \WIDTH задати необхідну ширину стовпців.

3. Клавішами управління курсором вибрати окремо кожен осередок і ввести заголовок таблиці, рамки, прізвища, стать.

4. Задати визначення осередків в стовпці "Нараховано".

#E3:#C3*#DS/30

#E4:#C4*#D4/30

#E5:#C5*#05/30

5. Ввести команду \TEXT.

6. Створити текстовим редактором "GURU" наступну програму:

```
CLEAR /*очистить экран*/
E.LSTR = 80 /*задати довжину символьного рядка*/
OUTPUT "запити для ЕВ за розрахунком заробітної платні"
OUTPUT
OUTPUT "введіть оклади тих, що працюють"
OUTPUT
INPUT #C3 WITH #A3+"-" USING"dddd"
INPUT #C4 WITH #A4+"-" USING"dddd"
INPUT #C5 WITH #A5+"-" USING"dddd"
OUTPUT
OUTPUT "Введіть число відпрацьованих днів"
OUTPUT
INPUT #D3 WITH #A3+"-" USING"dd"
INPUT #D4 WITH #A4+"-" USING"dd"
INPUT #D5 WITH #A5+"-" USING"dd"
CLEAR
```

Запам'ятати цю програму у файлі MY_PROGRAM.

7. Запустити цю програму командою PERFORM MY_PROGRAM і ввести необхідні дані.

8. Ввести команду \COMPUTE. В результаті в стовпці "Нараховано" будуть розраховані необхідні значення.

9. Зберегти ЕВ у файлі командою \SAVE TO MY_FILE.

10. Вийти командою \STOP.

11. Роздрукувати командою \DISPLAY.

ПІДГОТОВКА ДО ВИКОНАННЯ ЗАВДАННЯ

1. Дайте відповіді на контрольні питання.
2. Ознайомтеся з прикладом розв’язання типового завдання.
3. Згідно заданому варіанту розробіть структуру ЕВ.
4. Підготуйте програму, за допомогою якої можна ввести дані в ЕВ.

ПОРЯДОК ВИКОНАННЯ ЗАВДАННЯ

1. Пред'явите викладачу розроблену по заданому варіанту структуру ЕВ.
2. Увійдіть до командного режиму "GURU".
3. За допомогою команди "CALC < розмір ЕВ >" створіть ЕВ.
4. Опишіть структуру ЕВ (згідно заданому варіанту).
5. Створіть програму, яка вводить дані в ЕВ. Запустіть і перевірте її роботу.
6. На основі початкових даних розрахуйте ЕВ.
7. Роздрукуйте ЕВ.
8. Результати покажіть викладачу.

ЗВІТ ПРО ВИКОНАНЕ ЗАВДАННЯ ПОВИНЕН МІСТИТИ:

1) відповіді на контрольні питання;
2) структуру ЕВ;
3) послідовність дій для створення ЕВ;
4) текст програми;
5) роздруківку з отриманими результатами;
6) висновки про результати роботи, переваги і недоліки ЕВ.

ЗАВДАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Коли ви переходите дорогу, то порахуйте, скільки машин проїхало мимо. Створіть ЕВ, в якій будуть наступні дані:

Марка автомобіля	Швидкість	Годин в дорозі	Відстань
ЗАЗ	100	5	---
.....	...	…	...

Написати програму, яка для кожного автомобіля на основі відомої швидкості і часу розраховувала б відстань.

2. Припустимо, що магнітофон ремонтував майстер. Він може створити на основі результатів своєї роботи ЕВ наступної структури: тип магнітофона, несправність, кількість однотипних деталей для заміни, вартість однієї деталі, загальна вартість. Поле "Загальна вартість" визначається таким

чином: "кількість однотипних деталей * вартість однієї деталі". Написати програму, яка для кожного магнітофона розрахує загальну вартість ремонту.

3. При прийомі людини на роботу створюється ЕВ наступної структури:

Ф. І. Б.	Оклад	Податок (у %)	Сума за рік
Іванов І. І.	1000	10	-----
.....	...	...	....

де сума за рік = (оклад - оклад * податок) * 12. Написати програму, яка обчислює суму за рік.

4. Припустимо, що ви створюєте програму, яка управляє роботою АЗС. Кожна помилка в програмі дає певний економічний збиток. У програмі може бути декілька однотипних помилок. Створити ЕВ наступної структури:

Тип помилки	К-ть помилок	Збиток від однієї	Загальний збиток
Пропущений оператор	3	1000	------
..... .	...	.....	……

де Загальний збиток = к-ть помилок*збиток від 1 помилки. Написати програму, яка обчислює загальний збиток для різних типів помилок.

5. Припустимо, вам необхідно дізнатися, наскільки змінилася кількість опадів за поточний місяць в поточному році в порівнянні з минулим. Створіть ЕВ наступної структури.

Місяць	Поточний рік	Минулий рік	Зміна
Січень	100	120	----------
.....	...	...	.....

Зміна = поточний рік – минулий рік. Написати програму, яка обчислює всі зміни за рік.

6. Результати тестування дітей заносяться в ЕВ наступної структури:

Прізвище	Бал за складання	Бал за множення	Загальний бал
Іванов	5	3	----------
.....	...	...	...

Загальний бал = Бал за складання + Бал за множення. По загальному балу судять про рівень підготовки дитини. Написати програму, яка обчислює загальний бал по балам тестування.

7. На біржі здійснюється операція продажу товару. Створити ЕВ наступної структури:

Найменування товару	Ціна за 1 шт.	К-ть в партії	Загальна сума
Плита бетонна	10000	5	-------
.....	...	...	...

Загальна сума = Ціна за 1 шт. * К-ть в партії. Написати програму, яка обчислює загальну суму для декількох товарів.

8. Створити ЕВ, що порівнює рівень води в річці минулого року і в поточному, для того, щоб визначити - чи буде повінь чи ні?:

Місяць	Рівень минулого року	Рівень в поточному році	Зміна
Січень	220	250	-------
...	…	…	....

Зміна = Рівень в поточному році - Рівень в минулому році. Написати програму, яка обчислює зміну рівня води в річці.

9. Створити ЕВ, за допомогою якої можна визначити: скільки кілометрів проїде транспортний засіб?.

Трансп. засіб	Літрів на 100 км.	Місткість бака	Відстань
ВАЗ 2105	5	40	----------
....	…	…	.....

Відстань = (Місткість бака/Літрів на 100 км.)*100. Написати програму, яка обчислює відстань для декількох транспортних засобів.

10. Припустимо, що майстер ремонтував телевізор. Він може створити на основі результатів своєї роботи ЕВ наступної структури:

Тип TV	Несправність	Кількість однотипних деталей для заміни	Вартість однієї деталі	Загальна сума
"РУБІН"	згорів резистор	5	1	----------
....	.....	…	…	.....

Загальна сума = (Кількість однотипних деталей для заміни * Вартість однієї деталі). Написати програму, яка для кожного телевізора розраховує вартість ремонту.

МЕТОДИЧНІ ВКАЗІВКИ

до організації самостійної та індивідуальної роботи студентів з дисципліни

“МЕТОДИ ТА ЗАСОБИ ПОДАННЯ ЗНАНЬ”

ВСТУП

Курс “Методи та засоби подання знань” – обов'язковий компонент загальної та професійної освіти. Значення курсу “Методи та засоби подання знань” у загальноосвітній підготовці визначається насамперед тим, що комп’ютерні методи подання знань є фундаментом науково-дослідницької діяльності та науково-технічного прогресу.

Дисципліна “Методи та засоби подання знань” призначена для оволодіння майбутніми бакалаврами спеціальності ”Інтелектуальні системи прийняття рішень” основними моделями організації і подання знань в комп’ютеризованих системах, на базі яких провадиться подальше вивчення спеціальних дисциплін, пов’язаних з фаховою діяльністю

У результаті вивчення дисципліни студент повинен одержати фундаментальні теоретичні знання у галузі баз знань інтелектуальних систем і закріпи їх на практичних заняттях.

Програма курсу “Методи та засоби подання знань” охоплює достатній обсяг матеріалу, який дозволяє підготувати бакалаврів належного рівня. Вона складена згідно з вимогами “Положення про програму дисципліни” і є нормативним документом, який визначає мету і завдання курсу, місце курсу серед дисциплін професійної підготовки.

Типова програма відповідає діючим підручникам, що використовуються у навчальному процесі.

Вивчення навчальної дисципліни “Методи та засоби подання знань” спирається на такі дисципліни: „Основи дискретної математики”, "Основи програмування та алгоритмічні мови", "Теорія ймовірностей, імовірнісні процеси і математична статистика", "Об'єктно-орієнтоване програмування", "Системний аналіз та проектування систем обробки інформації", "Організація баз даних і знань".

Мета курсу полягає в тому щоб забезпечити загальний розвиток світогляду студентів, їх ознайомлення з методами та засобами подання знань у базі знань інтелектуальних систем прийняття рішень. Метою застосування баз знань до конкретної проблеми є підвищення ступеня обґрунтованості рішення, що приймається.

Досягнення цієї загальної мети у практиці викладення курсу можна здійснювати різними шляхами:

- конкретизацією теорій, явищ і процесів під час вивчення курсу та закріпленні знань, використовуючи навчальний матеріал предметів профциклу;
- показом практичного використання в даній професійній діяльності знань, отриманих під час вивчення курсу;
- складанням задач з професійно спрямованим змістом, виконанням при їх рішенні розрахунків, пов'язаних з майбутньою професійною діяльністю студентів;
- проведенням практичних робіт із курсу, інтегрованих з деякими практичними роботами профциклу;

- використанням діафільмів, кіно- і відеофільмів із загальноосвітніх дисциплін профциклу з ілюстрацією в них наступності та взаємозв'язку основ системного аналізу і професійних знань.

Завдання курсу полягає у тому, щоб навчити студентів системі методів та засобів подання знань у базі знань інтелектуальної системи.

Критерії оцінки успішності повинні відповідати навчальній програмі й найбільш важливим вимогам до знань студентів:

1. Знання фактів, явищ. Вірне, науково достовірне їх пояснення.
2. Оволодіння науковими термінами, поняттями, законами, методами, правилами; вміння користуватися ними при поясненні нових фактів, розв‘язуванні різних питань і виконанні практичних завдань.
3. Максимальна ясність, точність думки, вміння відстоювати свої погляди, захищати їх.

Методи і форми викладання дисципліни

Вивчення дисципліни передбачає лекційні та практичні заняття. Значна частина матеріалу дисципліни відведена під індивідуальні заняття студентів під керівництвом та СРС. СРС використовується студентами для проробки лекційного матеріалу, навчально-методичної літератури, при підготовці до практичних занять.

Навчальна дисципліна “Методи та засоби подання знань” розглядається як елемент загальної освіти і відіграє роль апарату вивчення та засвоєння закономірностей навколишнього світу та фундаменту професійних знань у галузі інтелектуальних систем. Навчання слід спрямувати на формування уміння використовувати наукові знання для розв'язання практичних завдань у професійній діяльності, на прийняття обґрунтованих рішень у конкретних виробничих ситуаціях, на застосування методів і теорій у поясненні суті інформаційних і технологічних процесів.

Форми і засоби проміжного та підсумкового контролю: експрес-контроль рівня готовності студента до проведення та практичних робіт; перевірка виконання позааудиторних завдань; оцінка роботи студента під час заняття (виступи, доповнення, участь у дискусії); виконання домашніх завдань; контрольні роботи в кінці залікового кредиту. Оцінка індивідуальних результатів здобуття знань студентами проводиться у формі заліку за кредитно-модульною методологією навчання, критерії якої визначаються у навчальній робочій програми за стобальною системою, яка трансформується у стандартні залікові диференційовані оцінки відповідно до вимог Міністерства освіти та науки України.

Форма підсумкового контролю – ПМК.

Загальний обсяг навчальної дисципліни - 108 годин, з них лекції: 36 годин, практичні: 34 години, самостійна робота: 11 годин, індивідуальні заняття: 27 годин.

ТЕОРЕТИЧНІ ОСНОВИ

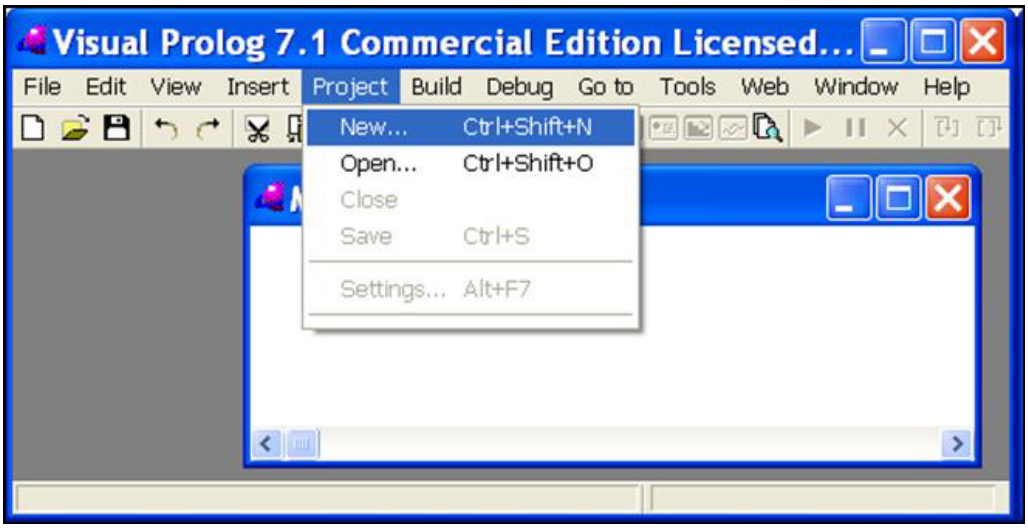
Формально-логічні методи та засоби подання знань

1. Створення проектів у Visual Prolog

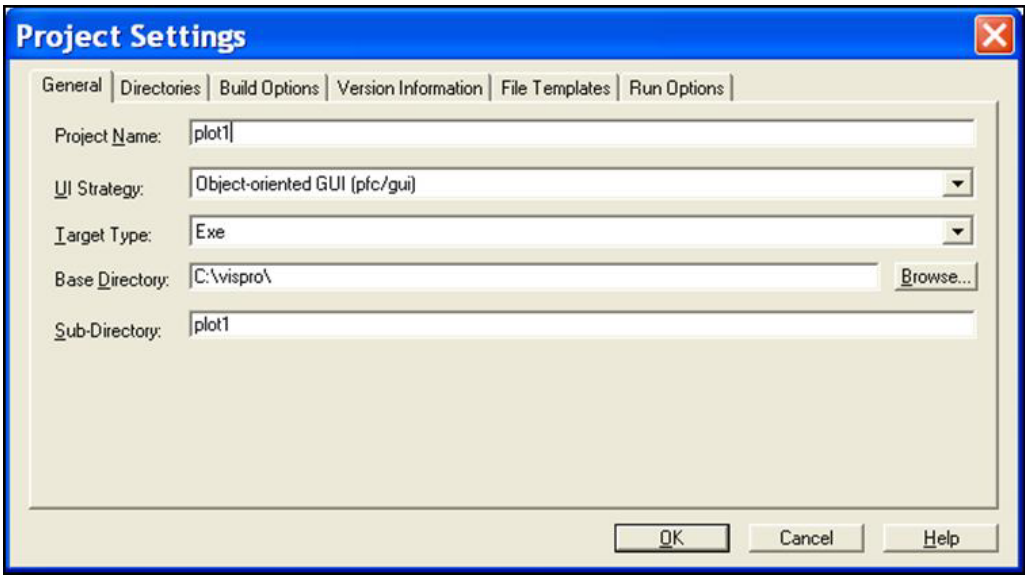
Середовище, у якому створюються проекти, називається IDE (скорочення від *Integrated Development Environment* - інтегроване середовище розробки). IDE Visual Prolog показана на мал. 1.1 (ми посилатимемося на меню IDE як на «меню завдань»). Система вікон і діалогів, яка створюється для спілкування з потенційними користувачами, називається *Graphical User Interface* - графічний інтерфейс або, скорочено, GUI.

1.1. *Створення нового проекту GUI: name*

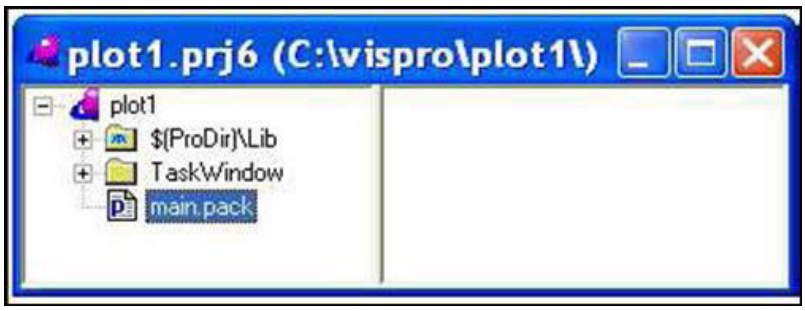
Цей крок досить простий. Виберіть команду *Project/new* з меню завдань, як показано на мал. 1.1. Потім заповните діалогове вікно *Project Settings,* як на мал. 1.2. Натисніть кнопку *Create,* і перед вами з'явиться вікно дерева проекту (мал. 1.3).



Мал. 1.1 Створення нового проекту



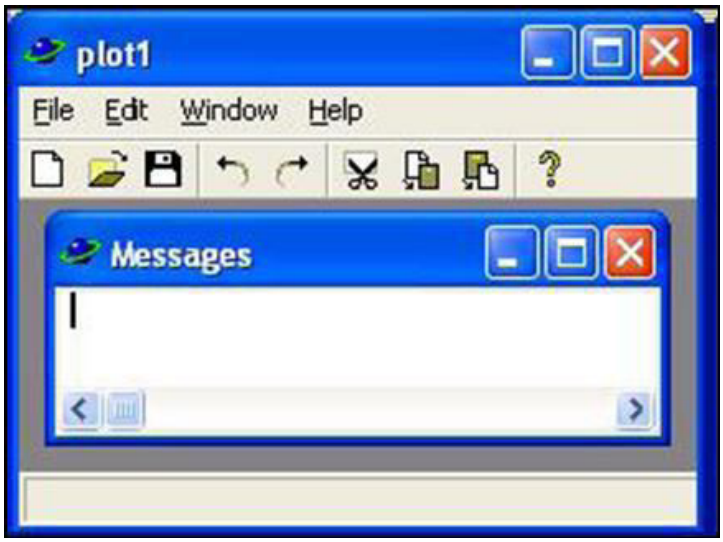
Мал. 1.2 Налаштування проекту



Мал. 1.3 Дерево проекту

1.2. *Компіляція і запуск програми* Для того, щоб відкомпілювати програму, виберіть команду *Build/build* з меню завдань, як показано на мал. 1.4. Для запуску програми виберіть *Build/execute* з панелі завдань, і на екрані з'явиться вікно, схоже на зображене на мал. 1.5. Для того, щоб вийти з програми, все, що від вас вимагається - клацнути по кнопці у вигляді X, яка знаходиться у верхньому правому кутку вікна; якщо вам так хочеться, виберіть пункт *File/exit* меню завдань додатку.

Мал. 1.4 Збірка проекту



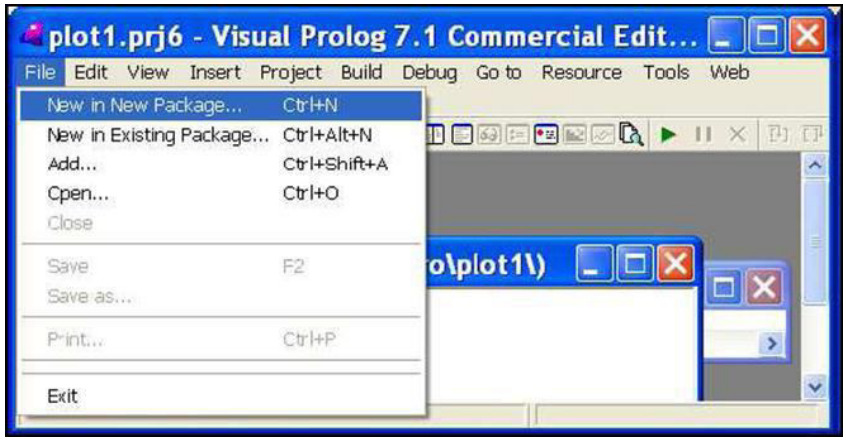
Мал. 1.5 Порожній додаток

2. Форми

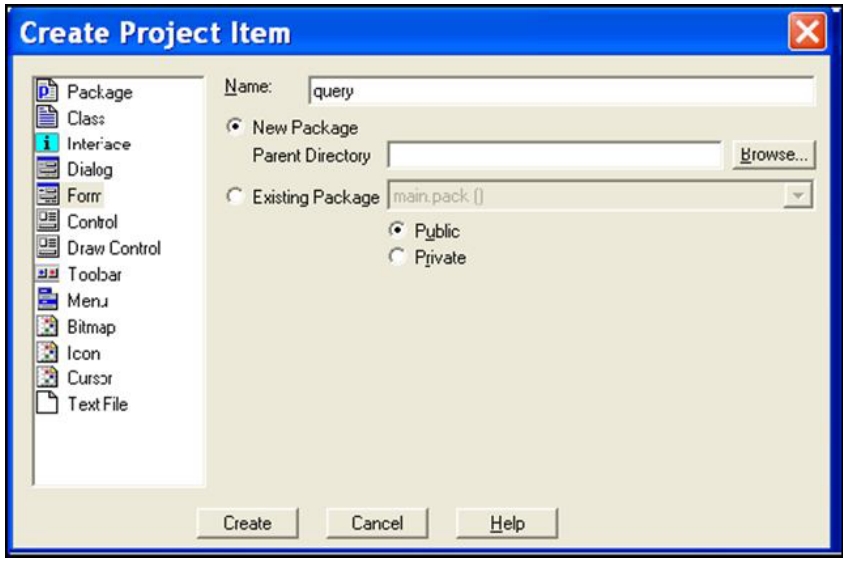
У цьому розділі додамо форму до порожнього проекту, який був створений в п. 1.1. **Форма** - це поверхня вікна, куди ви можете додавати компоненти на зразок **кнопок,** які запускають певні дії, **рядків введення**, які використовуються для введення тексту, і **полів для малювання**, де можна малювати.

2.1. *Створення форми: folder/name*

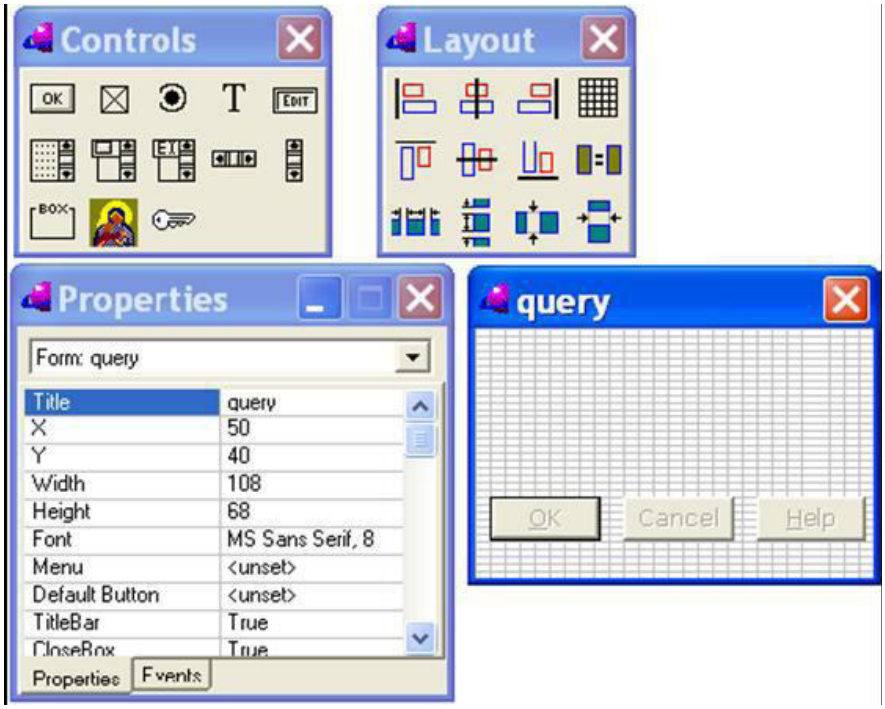
Щоб створити форму, виберіть команду *File/new in New Package* з меню завдань, як на мал. 2.1. Виберіть на лівій панелі пункт *Form* і заповните діалогове вікно *Create Project Item* відповідно мал. 2.2. Назва нової форми - *query*. Оскільки ми вибрали варіант New in New Package, Visual Prolog створить форму в пакеті з такою ж назвою. Після натиснення на кнопку Create діалогового вікна Create Project Item, IDE покаже вам прототип нової форми, як на мал. 2.3. Ви можете змінити розміри прямокутника вікна, зробивши його трохи більше прототипу. Щоб зробити це, натисніть мишкою на нижньому правому куті форми і тягніть його, як ви робите, коли змінюєте розміри звичайного вікна.



Мал. 2.1 Додавання нового предмету до дерева проекту

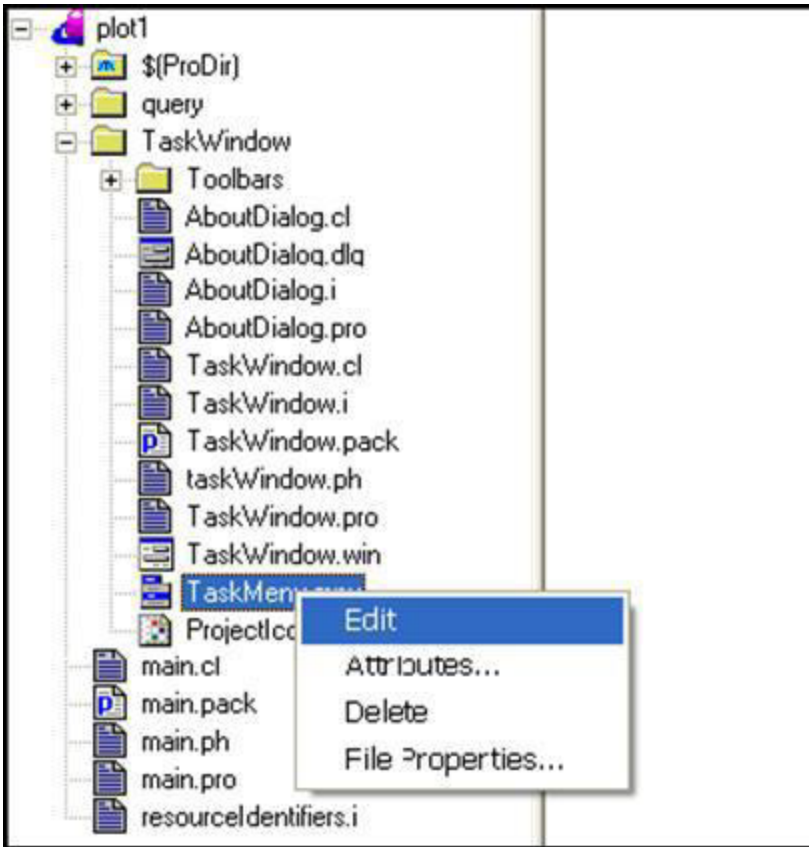


Мал. 2.2 Створення нової форми

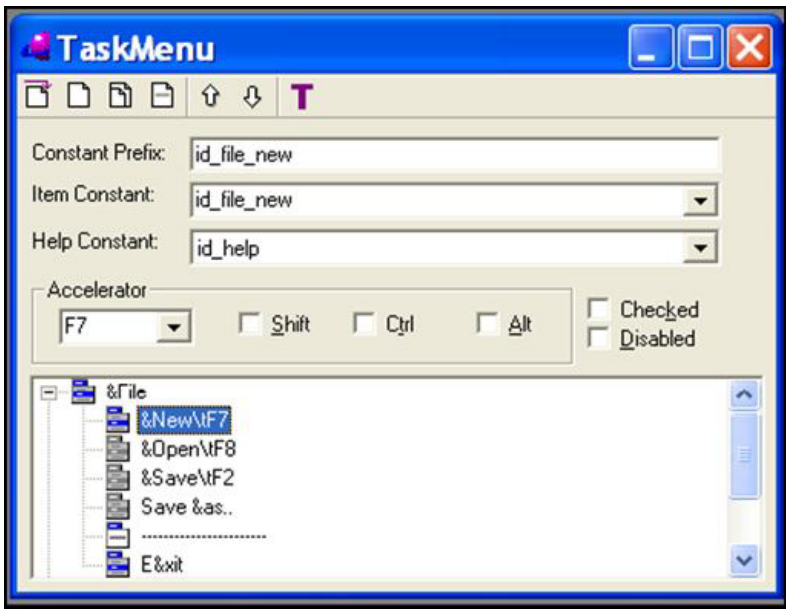


Мал. 2.3 Зміна розмірів форми

2.2. *Включення панелі завдань (пункт File/new)* Коли ми запускали порожній додаток, то пункт меню *File/new* був відключений. Щоб включити його, двічі клацніть по гілці дерева проекту під назвою *Taskme.mnu* (мал. 2.4). Потім, розверніть дерево, що знаходиться в нижній частині діалогового вікна *Taskmenu*, і приберіть галочку *Disabled*, що відноситься до пункту *&New/tF7*, як на мал. 2.5.

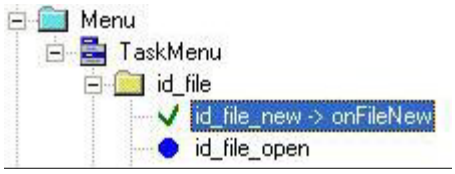


Мал. 2.4 Дерево проекту/меню завдань



Мал. 2.5 Включення елементу меню завдань

2.3. *Додавання коду у Експерт Коду до елементу дерева проекту* Щоб додати код до пункту *File/new*, натисніть на гілці дерева проекту *Taskwindow.win* правою кнопкою миші; перед вами відкриється контекстне меню. Виберіть пункт *Code Expert* (рис 2.6). Наслідуючі мал. 2.7, двічі клацніть на



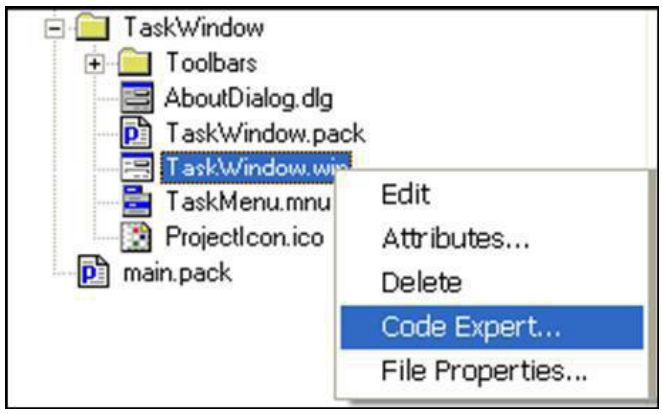
Нарешті, натисніть кнопку *Add* (орієнтуйтеся по мал. 2.7), і двічі клацніть по гілці *id_new->onFileNew*. Це відкриє текстовий редактор з наступним фрагментом коду:

```
clauses
        onFileNew(_Source, _MenuTag).
```

Побудуйте програму, а потім змініть цей фрагмент так:

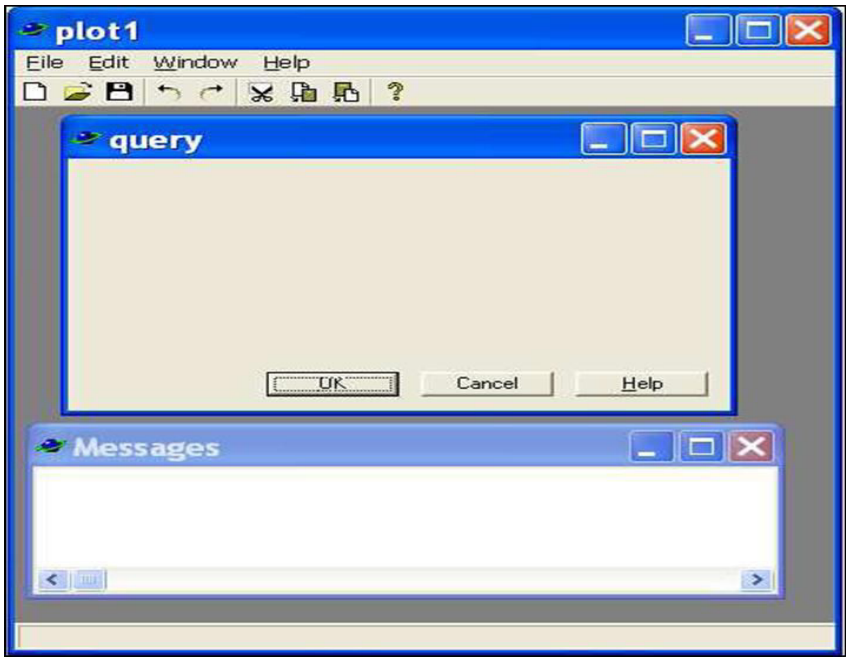
```
clauses
        onFileNew(W, _MenuTag) :-
                X = query::new(W),
                X:show().
```

Знову побудуйте програму, вибравши пункт *Build/build* на панелі завдань, згідно мал. 1.4. Запустивши програму, побачите, що, коли ви натискаєте *File/new*, створюється нова форма (мал. 2.8).



Мал. 2.6 Перехід в Code Expert

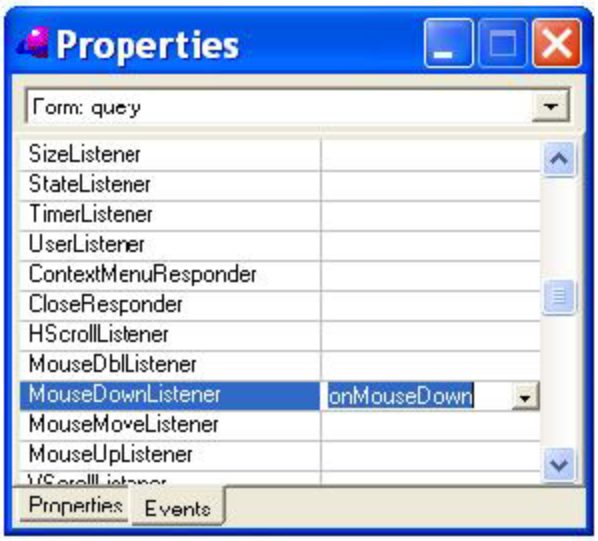
Мал. 2.7 Dialog and Window Expert



Мал. 2.8 Форма, що з'являється

3. Події миші

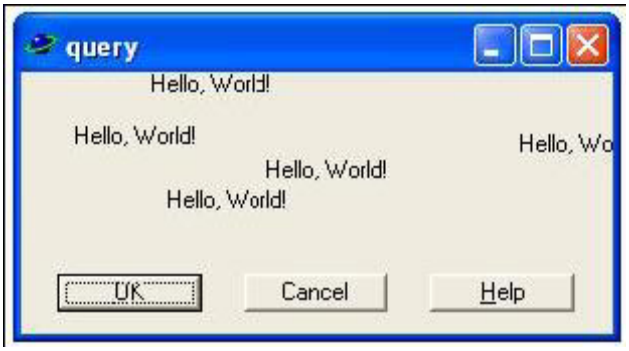
3.1. Додамо код до MouseDownListener. Натисніть на гілку *query.frm* у дереві проекту, щоб наново відкрити форму запитів, як показано на мал. 2.3; потім натисніть кнопку *Events* діалогового вікна *Properties*. Перед Вами відкриється список подій, як зображено нижче.



Двічі клацніть по події *MouseDownListener* і замініте процедуру onMouseDown/4 наступним кодом:

```
clauses
        onMouseDown(S, Point, _ShiftControlAlt, _Button) :-
                W= S:getVPIWindow(),
                Point= pnt(X, Y),
                vpi::drawText(W, X, Y, "Hello, World!").
```

Відкомпілюйте програму і запустить її. Виберіть *File/new,* щоб створити нову форму. Кожного разу, коли ви натискаєте де-небудь усередині на формі, програма розміщує на ній вітання.



3.2. *onPain.t* Ми використовували обробник події onMouseDown/4 для того, щоб намалювати що-небудь на формі. Є програмісти, які не вважають це за хорошу ідею. Насправді, існують мови, які роблять такий підхід дуже

важкоздійсненним. Java - це одна з таких мов; і, між іншим, вона при цьому дуже популярна. У Java будь-яке малювання слід проводити усередині наступного методу: *public abstract void doPaint(java.awt.Graphics g).*

Звичайно, це обмеження можна обійти, і навіть в Java. У будь-якому випадку, давайте навчимося малювати за допомогою обробника подій onPaint/3.

1. Створіть проект з ім'ям plot0.

2. Додайте нову форму query в новий пакет query.

3. Включите пункт меню завдань *File/new*, і додайте фрагмент

```
clauses
        onFileNew(S, _MenuTag) :-
                Q= query::new(S), Q:show().
```

к *ProjTree/TaskWindow.win/Menu/TaskMenu/id_file/id_file_new*.

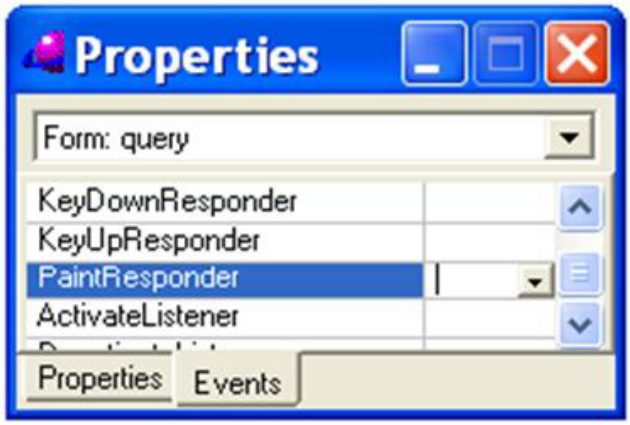
4. Побудуйте додаток. Наступний крок - додати фрагмент

```
class facts
mousePoint:pnt := pnt(-1, -1).
predicates onMouseDown : drawWindow::mouseDownListener.
clauses
onMouseDown(_S, Point, _ShiftControlAlt, _Button) :-
        mousePoint := Point,
        Point= pnt(X, Y),
        R= rct(X-8, Y-8, X+60, Y+8),
        invalidate(R).
```

в *MouseDownListener*. Тепер ми готові застосувати обробник події onPaint для форми query. onPaint буде викликаний кожного разу, коли прямокутна область на формі стає застарілою або невірною.

Коли зображення стає невірним? Коли вікно часткове або повністю перекрите іншим вікном, або коли ви оновлюєте його за допомогою предиката invalidate(R), де R – прямокутник. Прямокутник описується координатами його лівого верхнього і правого ніжнего кутів: R=rct(X-8, Y-8, X+60, Y+8). Ви, можливо, відмітили, що обробник події onMouseDown/4 робить невірною прямокутну область навколо місця клацання миші. В цьому випадку з цією «невірною» областю працюватиме обробник події onPaint, перемальовував її.

Щоб все це запрацювало, додайте фрагмент з мал. 3.2 в *PaintResponder*. У дереві проекту двічі клацніть по вітці query.frm. Натисніть на закладку Event діалогового вікна Properties (див. мал. 3.1 і 2.3); в списку подій, що з'явився, ви і знайдете *PaintResponder*. Цей новий підхід потрибує розуміння нових понять, таких, як конструкція if-then-else і глобальні змінні (о цих поняттях див. далі).



Мал. 3.1 PaintResponder

```
predicates
        onPaint : drawWindow::paintResponder.
clauses
        onPaint(_Source, _Rectangle, GDIObject) :-
                if pnt(X, Y)= mousePoint,
                X > 0 then
                        mousePoint := pnt(-1, -1),
                        GDIObject:drawText(pnt(X, Y), "Hello, World!", -
1)
                        else
                        succeed()
                end if.
```

Мал. 3.2 Код для PaintResponder

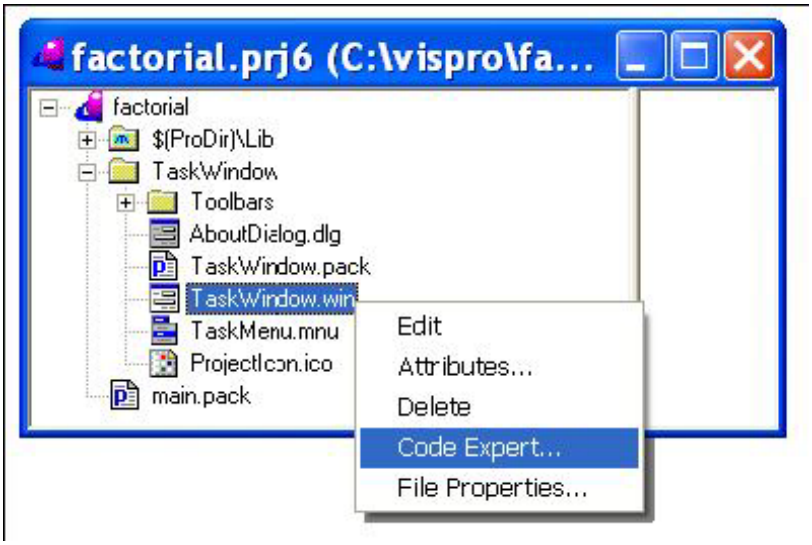
4. Методика створення проектів

4.1. Панель завдань. Меню завдань, показано на мал. 1.1 і 1.4, є головним меню Visual Prolog IDE. Запис *A/B* відноситься до одного з його пунктів. Наприклад, використовується команда *Project/New* для створення нового проекту (мал. 1.1). Воно [*діалогове вікно створення нового проекту*] має шість форм, а саме: *General, Directories, Build Options, Version Information, File Templates* і *Run Options*. В більшості випадків, нам потрібно буде заповнювати тільки форму *General*.

```
General
        Project Name: factorial
        UI Strategy: Object-oriented GUI(pfc/gui)
        Target Type: Exe
        Base Directory: C:\vispro
```

Коли ви знайдете крок, вказуючий: *Створіть новий GUI проект: factorial* вам слід увійти до діалогового вікна *Project Settings* (вибравши пункт *Project/New* з меню завдань VDE) і заповнити форму *General* як вказано вище.

4.2. Дерево проекту. Найлегший спосіб орієнтуватися у файлах і ресурсах - клацати мишкою по відповідних елементах дерева проекту:



Якщо ви двічі клацнете по папці, вона відкриється і покаже свій вміст. Якщо ви клацнете по елементу правою кнопкою миші, відкриється контекстне меню, як на малюнку. Щоб зайшлі в експерт коді і додали фрагмент коду в *Taskwindow.win* необхідно перейти до дерева проекту і зробити наступне:

1. Двічі клацніть по теці *TaskWindow*, щоб відкрити її, якщо вона закрита.

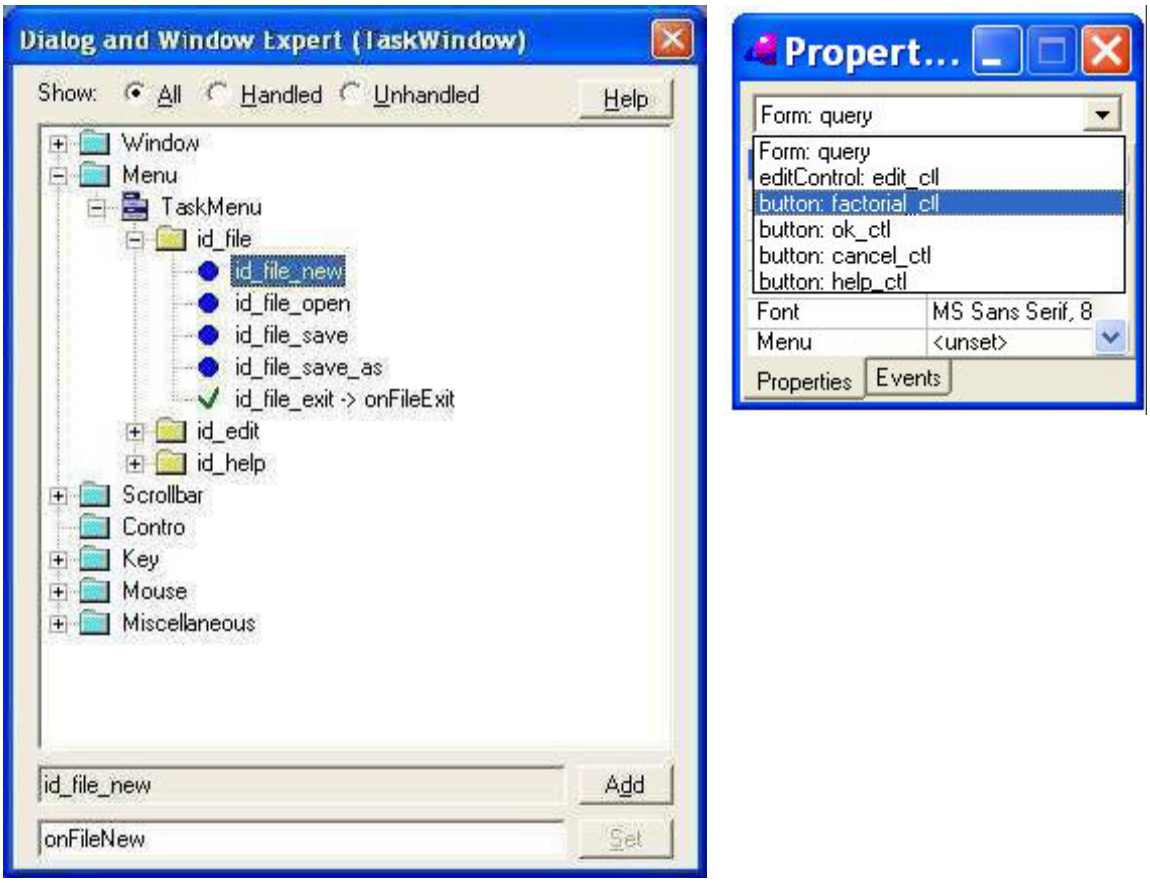
2. Клацніть правою кнопкою миші по гілці дерева проекту *TaskWindow.win*, щоб розкрити контекстне меню, в якому будуть наступні пункти: Edit, Attribute, Delete, Code Expert, File Properties...

3. Нарешті, виберіть пункт *Code Expert*

4.2.1. Code Expert. Експерт коду теж має форму дерева. Відповідно до своєї назви, експерт коду використовується для вставки коду в файли проекту. Щоб дістатися до експерта коду, ви повинні клацнути правою кнопкою миші по елементу дерева проекту, до якого ви хочете додати код. Потім, виберіть пункт *Code Expert* з контекстного меню. У випадку з формами, шлях до експерта коду можна скоротити.

Якщо ви двічі клацнете по гілці форми в дереві проекту, з'явиться прототип форми, разом з діалогом *Properties* і двома панелями компонентів, одна [**Controls**] для елементів управління, інша [**Layout**] для способів компоновки (див. мал. 2.3); якщо ви перейдете на закладку *Events* діалогового вікна *Properties*, ви отримаєте список подій для вибору. У попередніх розділах ми мали можливість працювати з двома обробниками подій, а саме з *MouseDownListener і* з *PaintResponder*.

На формах присутні такі компоненти, як кнопки і рядки введення. У верхній частині діалогового вікна *Properties* розташований список, що розкривається, для вибору цих компонентів; якщо ви виберете один з цих компонентів і перейдете на закладку *Events*, ви отримаєте список подій, відповідних вибраному компоненту.



Щоб переміщатися по дереву експерта коду і добиратися до точок, куди ви бажаєте вставити код, клацайте по потрібних гілках дерева. Якщо ви бажаєте, щоб експерт коду додав прототип до листка дерева, натисніть на цей листок і потім натисніть на кнопку *Add*, яка з'явиться внизу діалогового вікна. Потім двічі клацніть по листку знову, щоб перейти до коду.

Наприклад: у *Експерт коду додати код (розділ 2.3):*

```
clauses
        onFileNew(W, _MenuTag) :-
                S= query::new(W), S:show().
```

к TaskWindow.win/Code Expert/Menu/TaskMenu/id_file/id_file_new,
то необхідно зробити:

♣ ***Дерево проекту*** - Відкрийте теку *TaskWindow* дерева проекту, клацніть правою кнопкою миші по *TaskWindow.win*, щоб відкрити його контекстне меню, і виберіть пункт *Code Expert* (мал. 2.6).

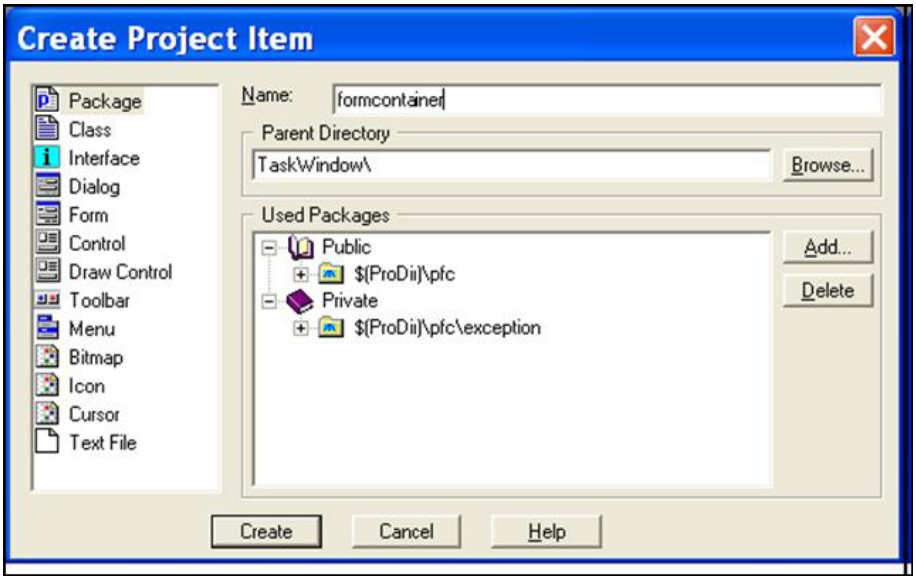
♣ ***Експерт коду*** - Двічі клацніть на *Menu→,* двічі клацніть на *TaskMenu→,* двічі клацніть на *id_file*. Виберіть *id_file_new* і натисніть кнопку *Add* для створення прототипу коду. Нарешті, двічі клацніть на *id_file_new→onFileNew*. Орієнтуйтеся по мал. 2.6, 2.7 і розділу 2.3. Додайте необхідний код до файлу TaskWindow.pro.

```
clauses
        onFileNew(W, _MenuTag) :- S= query::new(W), S:show().
```

4.3. Створення елементу проекту Щоб додати новий елемент до дерева проекту, виберіть пункт File/New in New Package з меню завдань,

якщо ви хочете вкласти елемент в новий пакет, названий так само, як і цей елемент. Якщо ви хочете вкласти елемент в існуючий пакет, виберіть пункт *File/New in Existing Package*. Стежите за тим, щоб розміщувати створювані елементи в потрібних теках. У наступному прикладі пакет, що містить форму query, розміщується в корені проекту factorial. Завжди вибирайте імена, які щось означають.

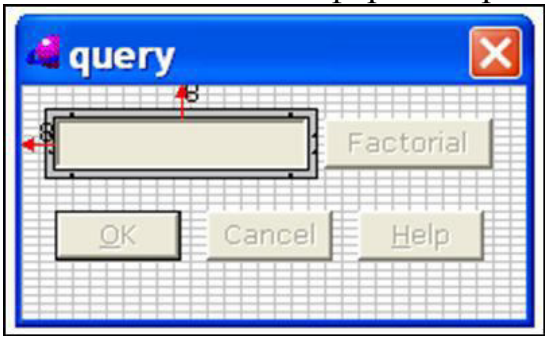
Наприклад, якщо ваш пакет містить комп'ютерну графіку, ви можете вибрати ім'я *canvasFolder1*; якщо він містить запити, хорошим ім'ям буде *formcontainer2*; і так далі. Приклад: ***Створіть новий пакет в корені дерева проекту*** factorial (мал. 4.1). Нехай ім'ям пакету буде formcontainer.



Мал 4.1 Створення нового пакету

Створіть нову форму (query) усередині пакету formcontainer, вибравши гілку, відповідну цьому пакету в дереві проекту, і вибравши пункт *File/New in Existing Package* з меню завдань (розділ 2.1).

Для того, щоб помістити вікно в цей пакет, переконаєтеся, що тека пакету вибрана, до того, як натискати *File/New in Existing Package*. Коли ви створюєте форму, IDE показує діалогове вікно *Form Window* (див. мал. 4.2). Ви можете скористатися можливістю змінити розміри вікна і додати рядок введення (під назвою edit_ctl) і кнопку (під назвою factorial_ctl), як показано на малюнку. Ви також можете дістатися до діалогового вікна *Form Window*, двічі клацнувши лівою кнопкою миші на ім'я форм в дереві проекту.



Мал. 4.2 Форма з рядком введення

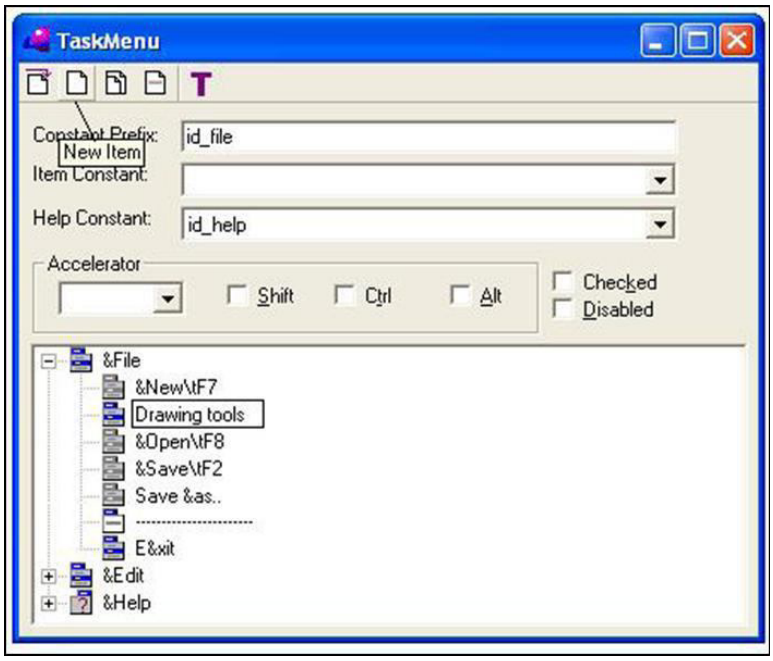
Якщо ви двічі клацнете по *ProjectTree/TaskMenu.mnu*, ви отримаєте діалогове вікно на зразок того, яке було показано на мал. 2.5. Ви можете розгортати дерево цієї специфікації меню, клацаючи по його вітках. Вам також може знадобитися включити який-небудь пункт меню, як в розділі 2.2. Також можливо створити новий елемент меню, як показано на нижчеприведеному малюнку, де натиснули на кнопку (ікону) під назвою *New Item* і створили пункт *Drawing Tools*, заповнивши діалогове вікно. Як ви можете побачити з малюнка, новий пункт створюється спочатку включеним. Символ & у записі *&File* позначає F як клавішу-акселератор.

Почнемо новий проект з нуля. Створіть новий ***проект GUI***: factorial.

1. ***Додайте новий пакет до дерева проекту***: factorial/formcontainer.

2. Створіть нову форму: forms/query (розділ 4.3). Додайте до неї *edit field* - рядок введення (з назвою edit_ctl) і кнопку (з назвою factorial_ctl), як на мал. 4.2.

3. Включите пункт *TaskMenu File/New* (розділ 2.2), потім додайте (розділ 2.3):




```
clauses
        onFileNew(W, _MenuTag) :-
                S= query::new(W), S:show().
```

к *TaskWindow.win/Menu/TaskMenu/id_file→id_file_new→onFileNew*.

4. Побудуйте ***додаток***, щоб вставити нові класи в проект.

4.4. Створення нового класу: folder/name. Щоб створити новий клас і вкласти його в пакет *formcontainer*, виберіть теку formcontainer в дереві проекту і виберіть пункт *File/New in Existing Package* з меню завдань VDE. Потім виберіть варіант *Class* в діалоговому вікні *Project Item* і введіть fn як ім'я класу в полі *Name*. Переконаєтеся, що прибрали галочку *Create Objects*.

Коли ви натиснете кнопку *Create*, Visual Prolog покаже файли fn.pro і fn.cl, які містять прототип класу fn. Наше завдання - додати функціональності до цих файлів, замінивши їх вміст лістингами, приведеними в розділі 4.6.

Потім побудуйте додаток ще раз, щоб переконатися, що Visual Prolog вставить клас fn в проект.

4.5. *Зміст* рядка введення. У Visual C, Delphi, Clean, і ін. досить складно отримати зміст рядка введення. Оцінюючи Visual Prolog, повинені сказати, що він пропонує найлегший доступ до елементів управління, в порівнянні з рештою мов.

IDE зберігає дескриптор вікна рядка введення у факт-змінної. Щоб оцінити всі переваги від цього, відкрийте query.frm, двічі клацнувши по його вітці в дереві проекту.

Виберіть button:factorial_ctl із списку діалогового вікна *Properties*, що розкривається, натисніть кнопку *Event*, і клацніть по строчці *ClickResponder* у списку подій; потім додайте наступний фрагмент до коду кнопки factorial_ctl:

```
clauses
        onFactorialClick(_S) = button::defaultAction() :-
                                fn::calculate(edit_ctl:getText()).
```

Побудуйте і запустите програму. У цьому розділі ми вивчили, як створити форму з кнопкою (factorial_ctl) і рядком введення (edit_ctl); також дізналися, як дістати доступ до змісту рядка введення. Нижче приведений клас fn, який реалізує функцію факторіалу.

```
%File: fn.cl
class fn
predicates
        classInfo : core::classInfo.
        calculate:(string) procedure.
end class fn

% File fn.pro
implement fn
        open core
class predicates
        fact:(integer, integer)
        procedure (i,o).
```

```
clauses
        classInfo("forms/fn", "1.0").

        fact(0, 1) :- !.
        fact(N, N*F) :- fact(N-1, F).
        calculate(X) :- N= toterm(X),
                fact(N, F), stdio::write(F, "\n").
end implement fn
```

Мал. 4.3 Клас fn

5.3. Рішення

Припустимо, що є предикат city(Name, Point), який визначає координати міста на карті. Предикат city/2 має домен

```
city : (string Name, pnt Position).
```

і може бути визначений як база фактів:

```
city("Salt Lake", pnt(30, 40)).
city("Logan", pnt(100, 120)).
city("Provo", pnt(100, 200)).
city("Yellowstone", pnt(200, 100)).
```

Цей предикат перевіряє, чи є задане положення заданого міста вірним. Ось декілька запитів, які можна поставити предикату city/2.

```
city("Salt Lake", pnt(30, 40)) → true
city("Logan", pnt(100, 200)) → false
city("Provo", pnt(100, 200)) → true
```

Немає сумніву, що ми зможете знайти застосування для подібного предиката. Проте, предикат, що повертає координати міста по заданій назві, може виявитися набагато кориснішим.

```
city(“Salt Lake”, P) → P = pnt(30, 40).
```

У цьому новому різновиді предикатів символи, що починаються із заголовної букви, називаються змінними. Приклади змінних: *X, Y, Wh, Who, B, A, Xs, Temperature, Humidity, Rate*. Таким чином, якщо ви хочете перевірити, чи є символ змінної, перевірте його першу букву. Якщо вона заголовна або навіть є знаком підкреслення (_), значить, ви маєте справу із змінною. Коли ви використовуєте змінну, як наприклад P в запиті city("Salt Lake", P), ви хочете знати, що потрібно підставити замість P, щоб зробити предикат city("Salt Lake", P) → true, тобто істинним. Відповіддю є P=pnt(30, 40). Визначимо предикат city/2.

Project *Settings*. Увійдіть до діалогового вікна *Project Settings,* вибравши пункт *Project/New* з меню завдань VDE, і заповните його.

General
Project Name: mapDataBase
UI Strategy: Object-oriented (pfc/GUI)
Target Type: Exe
Base Directory: C:\vispro
Sub-Directory: mapDataBase\

Create *Project Item: Form*. Виберіть кореневий вузол дерева проекту. Потім виберіть пункт меню *File/New*. У діалоговому вікні *Create Project Item*, виберіть варіант *form* і заповните діалогове вікно так: Name: map

Додайте наступні кнопки до нової форми: Logan, SaltLake, Provo. Window *Edit*. Змініте розміри нової форми. Вікно форми повинне мати достатній розмір, щоб відобразити нашу «карту». Залиште більше порожнього місця в центрі форми. Побудуйте *проект*. Це важливо: виберіть пункт *Build/Build* у меню завдань, інакше система видасть повідомлення про помилку на наступному кроці.

Project Tree/TaskMenu.mnu. Включите *File/New*.

Project *Tree/TaskWindow.win/Code Expert*. Додайте

```
clauses
        onFileNew(S, _MenuTag) :-
                X= map::new(S), X:show().
```

к *Menu/TaskMenu/id_file/id_file_new/onFileNew*. Знову виконаєте *Build/Build* для проекту (краще перестрахуватися, ніж потім жаліти).

Створіть клас. *Створіть* клас draw, як пояснювалося в розділі 4.4. Щоб створити новий клас, виділите корінь дерева проекту і виберіть пункт *File/New* меню задач IDE. Назва класу draw, і галочка *Create Objects* знята. Побудуйте проект, для того, щоб внести прототип нового класу до дерева проекту. Потім відредагуйте draw.cl і draw.pro як показано на мал. 5.2 і 5.3.

Для того, щоб викликати предикат drawThem кнопками, що позначають міста, відправляйтеся в дерево проекту і відкрийте форму map.frm, якщо вона ще не відкрита; у діалоговому вікні *Properties* виберіть із списку компонентів кнопку logan_ctl; перейдіть на закладку *Event* і додайте наступний фрагмент коду:

```
clauses
        onLoganClick(S) = button::defaultAction() :-
```

```
                Parent= S:getParent(),
                P= Parent:getVPIWindow(),
                draw::drawThem(P, "Logan").
```

к *ClickResponder*. Повторите дії для “Salt Lake” і “Provo”. Не забудьте замінити назви кнопок на logan_ctl, provo_ctl і saltlake_ctl відповідно; заміните також назву міста в drawThem з Logan на Provo або, відповідно, Salt Lake. Побудуйте проект і запустите програму.

Якщо ви не пам'ятаєте, як будувати і запускати програму, погляньте на розділ 1.2. У новому додатку виберіть пункт *File/New*. З'явиться нова форма. Коли ви натиснете на яку-небудь кнопку, програма намалює відповідне місто.

```
% File: draw.cl
class draw
        open core, vpiDomains
predicates
        classInfo : core::classInfo.
drawThem:(windowHandle, string) procedure.
end class draw
```

Мал. 5.2 mapDataBase/draw.cl

```
% File:draw.pro
implement draw
        open core, vpiDomains, vpi

constants
        className = "draw".
        classVersion = "".
class facts city:(string, pnt).

clauses
        classInfo(className, classVersion).
        city("Salt Lake", pnt(30, 40)).
        city("Logan", pnt(100, 120)).
        city("Provo", pnt(100, 80)).
        city("Yellowstone", pnt(200, 100)).
        drawThem(Win, Name) :-
                B= brush(pat_solid, color_red),
                winSetBrush(Win, B),
                city(Name, P), !, P= pnt(X1, Y1),
                X2= X1+20, Y2= Y1+20,
                drawEllipse(Win, rct(X1, Y1, X2, Y2)).
        drawThem(_Win, _Name).
end implement draw
```

Мал. 5.3 mapDataBase/draw.pro

5.4. Множинні рішення

У попередньому розділі ми бачили предикати, які повертали рішення через свої змінні, а не перевіряли, істинне відношення або помилково. У наведеному вище прикладі предикат city/2 використовувався для отримання тільки одного рішення. Проте, існують ситуації, що вимагають більшої кількості рішень.

Нехай conn/2 буде предикатом, що встановлює зв'язок між двома містами.

```
conn(pnt(30, 40), pnt(100, 120)).
conn(pnt(100, 120), pnt(100, 200)).
conn(pnt(30, 40), pnt(200, 100)). ...
```

Ви можете використовувати його для визначення всіх зв'язків між містами, як показано в прикладі:

```
conn(pnt(30, 40),        W).
        → W= pnt(100, 120)
        → W=pnt(200, 100)
conn(X, Y).
        → X=pnt(30, 40)/Y=pnt(100, 120)
        → X= pnt(100, 120)/Y=pnt(100, 200)
        → X=pnt(30, 40)/Y=pnt(200, 100))
```

Розглянемо запит: *conn(pnt(30, 40), W)?*

Відповіддю може бути W= pnt(100, 120), але їм також може бути W= pnt(200, 100).

5.4.1. Програма, що використовує предикати з множинними рішеннями Давайте створимо програму, щоб показати всю красу можливості множинних рішень в Пролозі - можливості, відсутній в інших мовах.

Project Settings. Зайдіть в діалогове вікно *Project Settings,* вибравши команду *Project/New*, і заповните його так:

Project Name: drawMap

UI Strategy: Object-oriented GUI (pfc/gui)

Create Project Item: Package. Виділите в дереві проекту вузол *drawMap*. Виберіть пункт *File/New in New Package*. У діалоговому вікні *Create Project Item* виберіть *package* і заповните поля: Name: plotter Parent Directory:

Create Project Item: Form. Виділите вузол *plotter* дерева проекту. Виберіть пункт *File/ New in New Package*. У діалоговому вікні *Create Project Item* виберіть пункт *Form*. Заповните поля:

Name: map

Package: plotter.pack (plotter\)

Включите форму в проект. Виберіть *Build/Build* у меню завдань.

Project Tree/TaskMenu.mnu.

Включите пункт *File/New.*

Project *Tree/TaskWindow.win/Code Expert*.

Додайте

```
clauses
        onFileNew(S, _MenuTag) :-   F= map::new(S), F:show().
```

к *Menu/TaskMenu/id_file/id_file_new/onFileNew*.

Створіть клас. *Створіть* клас *draw*, як пояснено в розділі 4.4. Приберіть галочку *Create Objects*. Внесіть код для класу у файли draw.cl і draw.pro як показано на мал. 5.4. Побудуйте додаток.

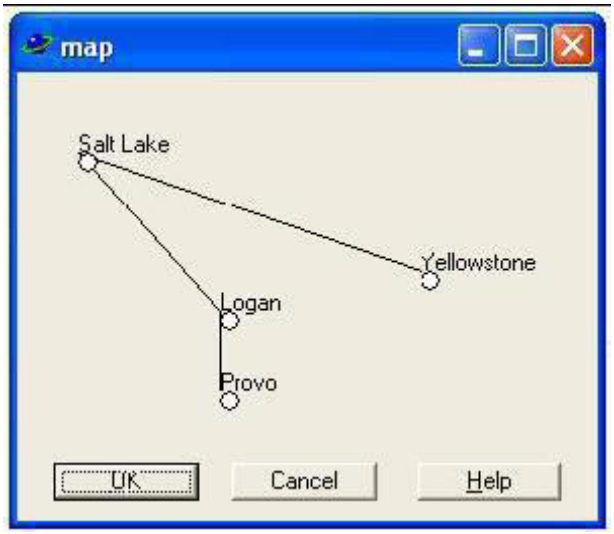
```
% File: draw.cl
class draw
        open core, vpiDomains
predicates
        drawThem:(windowHandle) procedure.
end class draw

% File: draw.pro
implement draw
        open core, vpiDomains, vpi
class facts
        city:(string Name, pnt Position).
        conn:(pnt, pnt).
class predicates
        connections:( windowHandle).
        drawCities:(windowHandle).
clauses
        city("Salt Lake", pnt(30, 40)).
        city("Logan", pnt(100, 120)).
        city("Provo", pnt(100, 160)).
        city("Yellowstone", pnt(200, 100)).
conn(pnt(30, 40), pnt(100, 120)).
conn(pnt(100, 120), pnt(100, 160)).
conn(pnt(30, 40), pnt(200, 100)).
drawCities(W) :- city(N, P), P= pnt(X1, Y1), X2= X1+10, Y2= Y1+10,
        drawEllipse(W, rct(X1, Y1, X2, Y2)),
        drawText(W, X1, Y1, N), fail.
drawCities(_Win).
connections(Win) :- conn(P1, P2), drawLine(Win, P1, P2), fail.
connections(_Win).
drawThem(Win) :- connections(Win), drawCities(Win).
end implement draw
```

Мал. 5.4 draw.cl и draw.pro

Project Tree/map.frm. Відкрийте map.frm і вставте наступний код в *PaintResponder*:

```
clauses
        onPaint(S, _Rectangle, _GDIObject) :-
                W= S:getVPIWindow(), draw::drawThem(W).
```

Якщо ви побудуєте і запустите програму, коли натиснете *File/New, то* повинні отримати вікно з картою, такою, як зображена на мал. 5.5



Мал. 5.5 Міста штату Юту

6. Пропозиції Хорна

Пропозицією Хорна може бути одиночний предикат. Наприклад, в нижченаведеному списку знаходяться чотири однопредікатних пропозиції Хорна.

```
city("Salt Lake", pnt(30, 40)).
city("Logan", pnt(100, 120)).
city("Provo", pnt(100, 200)).
city("Yellowstone", pnt(200, 100)).
```

Однопредикатні пропозиції Хорна називаються фактами. В даному прикладі факти встановлюють відношення між містом і його координатами. Доменом цих предикатів є множина пар, що складаються з назви міста і його координат. Пропозиція Хорна може також мати вид

```
H:-T1, T2, T3…,
```

де Ti і H - предикати. Так, запис

```
drawThem(Win):- connections(Win), drawCities(Win).
```

є прикладом пропозиції Хорна. У пропозиції Хорна частина, розташована до знаку :-, називається *head* - голова (заголовок). Частина, розташована після знаку:- називається *tail* - хвіст.

У даному прикладі головою є drawThem(Win), а хвостом – connections(Win), drawCities(Win). Сукупність пропозицій Хорна з однаковим заголовком визначає предикат.

Наприклад, чотири пропозиції Хорна про міста визначають предикат city/2, і

```
drawThem(Win) :- connections(Win), drawCities(Win).
```

визначає drawThem/1.

7. **Оголошення**

Визначення предиката не буде повним без оголошення типів і *flow pattern* - шаблону потоку. Ось оголошення типів і потоку (режиму) drawThem/1.

```
predicates
        drawThem : (windowHandle) procedure (i).
```

Оголошення типу свідчить, що аргумент Win предиката drawThem(Win) має тип windowHandle. Оголошення режиму стверджує, що аргумент drawThem/1 зобов'язаний бути вхідним, тобто, він є константою, а не вільною змінною.

В цьому випадку предикат приймає дані через свій аргумент. Зазвичай вам не потрібно надавати оголошення режиму. Компілятор самостійно зробить вивід про режими предиката. Якщо він не зможе це зробити, він видасть повідомлення про помилку, і ви зможете додати оголошення режиму самі. Якщо предикат передбачається використовувати тільки усередині його класу, він оголошується як class *predicate* - предикат класу. Наприклад:

```
class predicates
        connections : ( windowHandle ).
        drawCities : (windowHandle ).
```

Однопредикатні пропозиції Хорна можуть бути оголошені як факти. Наприклад:

```
class facts
        city : (string Name, pnt Position).
        conn : (pnt, pnt).
```

Пізніше ви побачите, що факти можуть бути додані (*asserted*) або видалені (*retracted*) з бази даних. Аргументи conn/2 - пара значень типу pnt. Тип pnt визначений в класі vpiDomains. Щоб використовувати його в класі draw, є дві можливості. Можна явно назвати клас, до якого належить pnt:

```
class facts
        conn : (vpiDomains::pnt, vpi::Domains::pnt).
```

або інакше можна відкрити цей клас усередині класу draw. Вибрано другий варіант: open core, vpiDomains, vpi

8. Оголошення режимів детермінізму

Щоб оголосити, має предикат одне рішення або декілька, використовуються наступні ключові слова:

determ. предикат може завершитися невдало (*fail*), або успішно (*succeed*), з одним рішенням.

procedure Предикати цього вигляду завжди завершуються успішно, і мають одне рішення. Предикати connections/1 и drawCities/1, визначені на мал. 5.4, є процедурами, і могли б бути оголошені так:

```
class predicates
        connections:(windowHandle) procedure (i).
        drawCities:(windowHandle) procedure (i).
```

multi. multi предикат не може завершитися невдало і має декілька рішень.

nondeterm. nondeterm предикат може завершитися невдало або успішно з множиною рішень. Факти city/2 і connection/2 обидва nondeterm і мають наступне оголошення:

```
class facts
        city:(string Name, pnt Position) nondeterm.
        conn:(pnt, pnt) nondeterm.
```

Якщо у предиката є декілька рішень, і одне з цих рішень не задовольняє предикату в кон'юнкції, Пролог робить backtrack - відкіт, і пропонує інше рішення в спробі задовольнити кон'юнкції.

Погляньте на пропозицію Хорна:

```
connections(Win) :- conn(P1, P2),
        drawLine(Win, P1, P2), fail.
connections(_Win).
```

nondeterm факт conn(P1, P2) надає дві точки, які використовуються процедурою drawline(Win, P1, P2) для малювання прямої лінії. Потім, Пролог пробує задовольнити предикату fail, який, відповідаючи своїй назві, завжди закінчується невдало. Отже, Пролог робить відкіт і пробує інше рішення, поки він не вичерпає всі можливості і не спробує другу пропозицію connection/1, яке завжди успішно.

9. Предикати малювання

Розглянемо предикати малювання. Для них потрібний дескриптор (*handle1)* вікна, яке підтримуватиме малюнки і креслення. От як ви можете отримати дескриптор:

```
clauses
        onPaint(S, _Rectangle, _GDIObject) :-
                W= S:getVPIWindow(), draw::drawThem(W).
```

Усередині класу draw передаватиметься дескриптор W. Наприклад, в пропозиції

```
drawThem(Win) :- connections(Win), drawCities(Win).
```

він передається в connections/1, де є першим аргументом drawLine/3:

```
connections(Win) :- conn(P1, P2), drawLine(Win, P1, P2), fail.
```

Предикат drawLine(Win, P1, P1) креслить лінію з P1 в P2 на вікні Win. Як ви знаєте, P1 і P2 - точки, такі, як pnt(10, 20). Інший предикат, з яким ви вже знайомі, це

```
drawEllipse(W, rct(X1, Y1, X2, Y2))
```

який креслить еліпс на вікні W. Еліпс вписаний в прямокутник rxt(X1, Y1, X2, Y2), де X1, Y1 - координати верхнього лівого кута, а X2, Y2 - координати нижнього правого кута.

10. GDI об'єкт

У останньому прикладі малювання проводилося з обробника події onPaint. Коли це відбувається, хорошою ідеєю може бути використання методів так званого *об'єкту GDI*. Наступний приклад показує, як це працює.

Project Settings. Створіть наступний проект:

Project Name: drawMapObj

UI Strategy: Object-oriented GUI (pfc/gui)

Target type: Exe
Base Directory: C:\vispro
Створіть пакет: plotter. Створіть форму усередині plotter: map.
Project Tree/TaskMenu.mnu. Включите *File/New*. В меню задач *Build/Build* додаток, для того, щоб включити форму map в проект.
Project Tree/TaskWindow.win/Code Expert. Додайте

```
clauses
        onFileNew(S, _MenuTag) :- F= map::new(S), F:show().
```

к *Menu/TaskMenu/id_file/id_file_new/onFileNew.*
Створіть клас. Створіть клас draw усередині пакету plotter. Не забудьте відключити "*Create ob-jects*". Ви знайдете нову версію класу draw на мал. 5.6. Побудуйте додаток. Додайте

```
clauses
onPaint(_S, _Rectangle, GDIObject) :- draw::drawThem(GDIObject).
```

к *PaintResponder*'у форми map.frm.

```
%File: draw.cl
class draw open core
predicates drawThem:(windowGDI).
end class draw

%File: draw.pro
implement draw
        open core, vpiDomains, vpi
class facts
        city:(string Name, pnt Position).
        conn:(pnt, pnt).
class predicates
        connections:( windowGDI).
        drawCities:(windowGDI).
clauses
        city("Salt Lake", pnt(30, 40)).
        city("Logan", pnt(100, 120)).
        city("Provo", pnt(100, 160)).
        conn(pnt(30, 40) , pnt(100, 120)).
        conn(pnt(100, 120), pnt(100, 160)).
        drawCities(W) :- city(N, P), P= pnt(X1, Y1),
        X2= X1+10, Y2= Y1+10, W:drawEllipse(rct(X1, Y1, X2, Y2)),
        W:drawText(pnt(X1, Y1), N), fail.
        drawCities(_Win).
        connections(W) :- conn(P1, P2),
                W:drawLine(P1, P2), fail.
        connections(_W).
drawThem(Win) :- connections(Win), drawCities(Win).
end implement draw
```

Мал. 5.6 Клас draw.

11. **Рекурсія**. Схема, що використовується для обчислення суми елементів списку, називається редукцією, тому що вона скорочує розмірність вхідних даних. Фактично, списки можна вважати за одновимірні дані, а суму - величиною нульовій розмірності. Є два способи виконання скорочення: рекурсивний і ітеративний.

```
%Recursive reduction
class predicates
        sum:(real*, real) procedure (i, o).
clauses
        sum([], 0) :- !.
        sum([X|Xs], S+X) :- sum(Xs, S).
```

Термін ітеративний засновано на латинському слові iterum, яке означає знову. Ви також можете уявити, що він сходить до iter/itineris - шлях, дорога. Ітеративна програма повторюється знову і знову.

```
%Iterative reduction
red([], A, A) :- !.
red([X|Xs], A, S) :- red(Xs, X+A, S).
sumation(Xs, S) :- red(Xs, 0.0, S).
```

Припустимо, що замість складання ви хочете провести множення. В цьому випадку ви можете написати наступну програму:

```
%Iterative reduction
red([], A, A) :- !.
red([X|Xs], A, S) :- red(Xs, X*A, S).
product(Xs, S) :- red(Xs, 0.0, S).
```

Ці два шматочки коду ідентичні, за винятком однієї деталі: у одному з них в другому аргументі ітеративного виклику предиката red/3 знаходиться вираз X+A, тоді як в іншому знаходиться X*A. [Wadler & Bird] і [John Hughes] запропонували використання такого роду схожості в предикатах і функціях вищих порядків. Давайте подивимося, як можна використовувати цей вигідний метод програмування в Visual Prolog.

У програмі, зображеній на мал. 1, оголошено домен, який складається з двомісних функцій. У розділі, що містить предикати класу, визначені деякі функції, що належать цьому домену. Нарешті, визначимо предикат red/4, який зберігає структуру операції редукції.

```
implement main
        open core
domains
        pp2 = (real Argument1, real Argument2) -> real ReturnType.
clauses
        classInfo("main", "hi_order").
class predicates
        sum:pp2.
        prod:pp2.
        red:(pp2, real*, real, real) procedure (i, i, i, o).
clauses
```

```
        sum(X, Y)= Z :- Z=X+Y.
        prod(X, Y)= Z :- Z=X*Y.
        red(_P, [], A, A) :- !.
        red(P, [X|Xs], A, Ans) :- red(P, Xs, P(X, A), Ans).
        run():- console::init(),
        red(prod, [1,2,3,4,5], 1, X),
        stdio::write(X), stdio::nl,
        succeed().
end implement main
goal
mainExe::run(main::run).
```

Мал. 1: Скорочення in Prolog

Наступний приклад показує застосування схеми з подальшим кроком редукції для обчислення скалярного добутку двох списків.

```
class predicates
        dotproduct:(real*, real*, real) procedure (i, i, o).
clauses
        dotproduct([], _, 0) :- !.
        dotproduct(_, [], 0) :- !.
        dotproduct([X|Xs], [Y|Ys], X*Y+R) :- dotproduct(Xs, Ys, R).
```

Ми можемо застосувати цю схему в предикаті вищого порядку точно так, як і в схемы редукції. На мал. 2 ви можете знайти (не дуже ефективну) реалізацію zip в Пролозі.

Предикат zip має два аргументи. Перший аргумент є двомісною функцією; у лістингу 2 є дві з таких функцій, але ви можете визначити ще. Другий і третій аргументи містять списки. Четвертий аргумент містить результат операції. Визначення zip/4 просте:

```
zip(_P, [], _, []) :- !.
zip(_P, _, [], []) :- !.
zip(P, [X|Xs], [Y|Ys], [Z|As]) :- Z= P(X, Y),
zip(P, Xs, Ys, As).
```

Якщо який-небудь з вхідних списків спустіє, перша або друга пропозиція зупинить рекурсію, провівши порожній список як результат; інакше, третя пропозиція зчепить голови двох вхідних списків і продовжить обробку їх частин, що залишилися.

```
implement main
        open core
domains
        pp2 = (real Argument1, real Argument2) -> real ReturnType.
clauses
        classInfo("main", "zip").
        class predicates
        sum:pp2.
        prod:pp2.
```

```
        red:(pp2, real*, real, real) procedure (i, i, i, o).
        zip:(pp2, real*, real*, real*) procedure (i, i, i, o).
        dotprod:(real*, real*, real) procedure (i, i, o).
clauses
        sum(X, Y)= Z :- Z=X+Y.
        prod(X, Y)= Z :- Z=X*Y.
        zip(_P, [], _, []) :- !.
        zip(_P, _, [], []) :- !.
        zip(P, [X|Xs], [Y|Ys], [Z|As]) :- Z= P(X, Y), zip(P, Xs, Ys, As).
        red(_P, [], A, A) :- !.
        red(P, [X|Xs], A, Ans) :- red(P, Xs, P(X, A), Ans).
        dotprod(Xs, Ys, D) :- zip(prod, Xs, Ys, Z), red(sum, Z, 0, D).
        run():- console::init(),
        V1= [1,2,3,4,5], V2= [2,2,2,2,2],
        dotprod( V1, V2, X),
        stdio::write(X), stdio::nl.
end implement main
goal
mainExe::run(main::run).
```

Мал. 2: Zip in Prolog

12. Операції з рядками. Нижче ви знайдете декілька прикладів операцій з рядками. Цих прикладів повинно бути досить, щоб показати вам, як розбиратися з подібними структурами даних. Ви можете знайти більше операцій, використовуючи довідкову систему Visual Prolog.

```
implement main
open core, string
class predicates
        tokenize:(string, string_list) procedure (i, o).
clauses
        classInfo("main", "string_example").
        tokenize(S, [T|Ts]) :-
        frontToken(S, T, R), !, tokenize(R, Ts).
        tokenize(_, []).
        run():- console::init(),
        L= ["it ", "was ", "in ", "the ",
        "bleak ", "December!"],
        S= concatList(L), UCase= toUpperCase(S),
        RR= string::concat("case: ", Ucase),
        R1= string::concat("It is ", "upper ", RR),
        stdio::write(R1), stdio::nl, tokenize(R1, Us),
        stdio::write(Us), stdio::nl.
end implement main
goal
mainExe::run(main::run).
```

Спробуйте зрозуміти, як працює fronttoken, тому що це вельми корисний предикат. Він використовується для розбиття рядка на токени (token), в процесі, званому лексичним аналізом. [Так само, як в українській мові речення складається з окремих слів-«лексем», рядок коду складається із знаків - «токенів»]. Наприклад, якщо ви виконаєте frontToken(“break December”, T, R), то отримаєте T=”break”, и R=” December”.

Часто потрібно трансформувати строкове представлення терма в придатне для роботи. В цьому випадку, можна використовувати функцію toterm. У прикладі нижче Sx=stdio::readline() читає рядок з командного рядка в Sx; потім toterm трансформує строкове представленя в дійсне число; виклик hasdomain(real, IX) засвідчить, що toterm поверне дійсне число.

```
implement main
clauses
        classinfo("main", "toTerm_example").
        run():- console::init(),
        Sx= stdio::readLine(),
        hasdomain(real, IX),
        IX= toTerm(Sx),
        stdio::write(IX^2).
end implement main
goal mainExe::run(main::run).
```

Нижче ви зможете побачити, що ця схема працює також для списків і інших складених структур даних. На жаль, hasdomain не приймає знак зірочки „*“ в імені домена. Таким чином, ви повинні оголосити домен списку самостійно, або використовувати заздалегідь оголошений домен, такий, як core::integer_list.

```
implement main
clauses
        classInfo("main", "hasdomain_example").
        run():- console::init(),
        hasdomain(core::integer_list, Xs),
        Xs= toTerm(stdio::readLine()),
        stdio::write(Xs), stdio::nl.
end implement main
goal mainExe::run(main::run).
```

Ви навчилися перетворювати строкове представлення типів даних в терми Прологу. Тепер, давайте подивимося, як робити зворотну операцію, тобто перетворювати терм в його строкове представлення. Якщо вам потрібний стандартний вид терма, то ви можете використовувати предикат tostring.

```
implement main
clauses
        classInfo("main", "toString_example").
        run():- console::init(),
        Xs= [3.4, 2, 5, 8],
```

```
        Str= toString(Xs),
        Msg= string::concat("String representation: ", Str),
        stdio::write(Msg), stdio::nl.
end implement main
goal mainExe::run(main::run).
```

Якщо ви хочете красивіше відформатувати числа або інші прості терми при перетворенні їх в рядок, ви можете використовувати функцію форматування string::format; приклад приведений нижче.

```
implement main
clauses
        classInfo("main", "formatting").
        run():- console::init(),
        Str1= string::format("%8.3f\n %10.1e\n", 3.45678, 35678.0),
        Str2= string::format("%d\n %10d\n", 456, 18),
        Str3= string::format("%-10d\n %010d\n", 18, 18),
        stdio::write(Str1, Str2, Str3).
end implement main
goal mainExe::run(main::run).
```

У наведеному вище прикладі формат "%8.3f\n означає, що я хочу вивести на екран дійсне число і повернення каретки; ширина поля, відведеного для числа, рівна 8 знаків і число має бути відображене з трьома знаками після коми. Формат "%010d\n" вимагає цілого числа, вирівняного по правому краю поля шириною 10; порожні місця поля мають бути заповнені нулями. Формат “%-10d\n” визначає відображення цілого числа, вирівняного по лівому краю поля шириною 10; знак мінус указує на ліве вирівнювання; праве вирівнювання є установкою за умовчанням. Формат “%10.1e\n” визначає науковий запис для дійсних чисел. Нижче ви можете побачити те, що приклад виведе на екран. Ще нижче ви знайдете список типів даних, що приймаються рядком формату. Відмітьте, що, коли в текстове відображення конвертуються дійсні числа, вони обрізаються і округляються до довжини в 17 цифр, якщо не була вказана інша точність.

f Форматувати як дійсне число з фіксованою комою (як в 123.4).

e Форматувати дійсне число в експоненціальному записі (як в 1.234e+002).

g Форматувати у форматі f або e, де виходить коротше.

d Форматувати як знакове ціле.

u Форматувати як беззнакове ціле.

x Форматувати як шістнадцятирічне число.

n Форматувати як вісімкове число.

c Форматувати як символ.

b Форматувати як двійковий тип Visual Prolog.

r Форматувати як номер посилання бази даних.

p Форматувати як параметр процедури.

s Форматувати як рядок.

Корисні предикати для роботи з рядками. Нижче ви знайдете список інших строкових предикатів, які ви можете знайти корисними в розробці ваших застосувань; adjustBehaviour, adjustSide і caseSensivity слід вважати за тих, що мають наступні визначення:

```
domains
adjustBehaviour =
expand();
cutRear();
cutOpposite().
adjustSide = left(); right().
caseSensitivity =
caseSensitive();
caseInsensitive();
casePreserve().
adjust : (string Src, charCount FieldSize, adjustSide Side) ->
string AdjustedString.
adjust : (string Src, charCount FieldSize, string Padding,
adjustSide Side) -> string AdjustedString.
adjustLeft : (string Src, charCount FieldSize) ->
string AdjustedString.
adjustLeft : (string Src, charCount FieldSize,
string Padding) -> string AdjustedString.
adjustRight : (string Src, charCount FieldSize) ->
string AdjustedString.
adjustRight : (string Src, charCount FieldSize,
s tring Padding) -> string AdjustedString.
adjustRight : (string Src, charCount FieldSize, string Padding,
adjustBehaviour Behaviour) -> string AdjustedString.


/**********************************
Project Name: adjust_example
UI Strategy: console
**********************************/
implement main
clauses
        classInfo("main", "adjust_example").
        run():- console::init(),
```

```
FstLn=
"Rage -- Goddess, sing the rage of Peleus' son Achilles,",
Str= string::adjust(FstLn, 60, "*", string::right),
stdio::write(Str), stdio::nl.
end implement main
goal mainExe::run(main::run).
concat : (string First, string Last) ->
string Output procedure (i,i).
concatList : (core::string_list Source) ->
string Output procedure (i).
concatWithDelimiter : (core::string_list Source,
string Delimiter) ->
string Output procedure (i,i).
create : (charCount Length) -> string Output procedure (i).
create : (charCount Length, string Fill) ->
string Output procedure (i,i).
createFromCharList : (core::char_list CharList) ->
string String.


/**********************************
Project Name: stringops
UI Strategy: console
**********************************/
implement main
clauses
classInfo("main", "stringops").
run():-
console::init(),
SndLn= [ "murderous", "doomed",
"that cost the Achaeans countless losses"],
Str= string::concatWithDelimiter(SndLn, ", "),
Rule= string::create(20, "-"),
stdio::write(Str), stdio::nl,
stdio::write(Rule), stdio::nl.
end implement main
goal mainExe::run(main::run).
equalIgnoreCase : (string First, string Second) determ (i,i).
front : (string Source, charCount Position,
```

```
string First, string Last) procedure (i,i,o,o).
frontChar : (string Source, char First, string Last) determ (i,o,o).
frontToken : (string Source, string Token,
string Remainder) determ (i,o,o).
getCharFromValue : (core::unsigned16 Value) -> char Char.
getCharValue : (char Char) -> core::unsigned16 Value.
hasAlpha : (string Source) determ (i).
hasDecimalDigits : (string Source) determ (i).
hasPrefix : (string Source, string Prefix,
string Rest) determ (i,i,o).
hasSuffix : (string Source, string Suffix,
string Rest) determ (i,i,o).
isLowerCase : (string Source) determ (i).
isUpperCase : (string Source) determ (i).
isWhiteSpace : (string Source) determ.
lastChar : (string Source, string First,
char Last) determ (i,o,o).
length : (string Source) ->
charCount Length procedure (i).
replace : (string Source, string ReplaceWhat,
string ReplaceWith, caseSensitivity Case) ->
string Output procedure
replaceAll : (string Source, string ReplaceWhat,
string ReplaceWith) -> string Output.
replaceAll : (string Source, string ReplaceWhat,
string ReplaceWith,
caseSensitivity Case) ->
string Output.
search : (string Source, string LookFor) ->
charCount Position determ (i,i).
search : (string Source, string LookFor,
caseSensitivity Case) ->
charCount Position determ (i,i,i).
split : (string Input, string Separators) -> string_list.
split_delimiter : (string Source, string Delimiter) ->
core::string_list List procedure (i,i).
subString : (string Source, charCount Position,
charCount HowLong) ->
```

```
string Output procedure (i,i,i).
toLowerCase : (string Source) ->
string Output procedure (i).
/* Convert string characters to lowercase */
toUpperCase : (string Source) ->
string Output procedure (i).
/* Convert string character to uppercase */
trim : (string Source) ->
string Output procedure (i).
/* Removes both leading and trailing
whitespaces from the string. */
trimFront : (string Source) ->
string Output procedure (i).
/* Removes leading whitespaces from the string. */
trimInner : (string Source) ->
string Output procedure (i).
/* Removes groups of whitespaces from the Source line. */
trimRear : (string Source) ->
string Output procedure (i).
/*Removes trailing whitespaces from the string. */


/**********************************
Project Name: trim_example
UI Strategy: console
**********************************/
implement main
clauses
classInfo("main", "trim_example").
run():-
console::init(),
Str= " murderous, doomed ",
T= string::trim(Str),
stdio::write(T), stdio::nl,
T1= string::trimInner(T),
stdio::write(T1), stdio::nl.
end implement main
goal mainExe::run(main::run).
```

13. **Оператори малювання.** У цьому розділі ми навчимося малювати, використовуючи обробник події **onPaint.** Почніть із створення проекту з формою, на якій ви малюватимете.

Створіть проект:

Project Name: painting
Object-oriented GUI (pfc/gui)

- Створіть пакет paintpack, приєднаний до Кореня дерева проекту.
- Вкладіть canvas.frm всередину paintpack. *Build/build* додаток.
- Введіть пункт *File/new* панелі завдань додатку, і додайте

```
clauses
        onFileNew(S, _MenuTag) :- X= canvas::new(S), X:show().
```

до *TaskWindow/Code Expert/Menu/TaskMenu/id_file/id_file_new*. Після цього кроку, якщо ви відкомпілюєте і запустите програму, то коли ви виберете пункт *File/new* в меню вашого додатку, буде виконаний наведений вище предикат.

Команда

X=canvas::new(S)

створює новий об'єкт X класу *window*. Це вікно буде дочернім вікну завдань S. Команда

X:show()

посилає повідомлення об'єкту X показати це вікно. Коли вікну потрібне перемальовування, воно викликає обробник події *onPainting*. Тому, якщо ви додасте інструкції до *onPainting*, вони будуть виконані.

Створіть клас dopaint усередині paintPack. Відключите Creates Objects. Додайте приведений нижче код до dopaint.cl и dopaint.pro.

```
% File dopaint.cl
class dopaint
        open core
predicates
        bkg:(windowGDI).
end class dopaint


%File: dopaint.pro
implement dopaint
        open core, vpiDomains
clauses
        bkg(W) :- P= [pnt(0,0), pnt(10,80), pnt(150, 100)],
                W:drawPolygon(P).
end implement dopaint
```

Build/Build додаток, щоб додати клас dopaint до проекту. Відправляйтеся в діалогове вікно canvas.frm, перейдіть на закладку *Events* і додайте наступний код до *PaintResponder*:

```
onPaint(_Source, _Rectangle, GDIObj) :- dopaint::bkg(GDIObj).
```

Якщо файл canvas.frm закритий, клацніть по його вітці в дереві проекту. Коли вікну потрібне промальовування, воно викликає onPaint, який в даному випадку викличе dopaint::bkg(GDIObj).

У класі dopaint є метод bkg(GDIObj), який малює на вікні, на яке указує аргумент GDIObj. Давайте зрозуміємо, що робить метод bkg(W). Як ви можете бачити, P зберігає список крапок. Кожна крапка визначена своїми координатами. Наприклад, pnt(0, 0) це крапка в координатах (0, 0).

Коли bkg(W) викликає W:drawPolygon(P), воно посилає повідомлення об'єкту W, запрошуючи у нього намалювати багатокутник, вершини якого задані списком P.

Відкомпілюйте програму, і перевірте, чи буде вона працювати в точності, як сказано. Давайте поліпшимо метод bkg(W). Він малює білий трикутник на поверхні canvas. Щоб зробити трикутник червоним, потрібно змінити кисть малювання. Кистю є двох аргументний об'єкт

```
brush = brush( patStyle PatStyle, color Color).
```

Кольори представлені числами, що визначають кількість червоного, зеленого і синього. Можна отримати багато кольорів, комбінуючи червону, зелену і синю палітри. Давайте представимо числа в шістнадцятирічному базисі.

Шістнадцятирічні числа мають шістнадцять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Червоний колір задається числом 0x000000FF, де 0x показує, що ми маємо справу з шістнадцятирічним базисом.

Представляти кольори числами непогано, але ви також можете використовувати словесні назви. Наприклад color_Red представляє червоний колір. Ось назви для інших шаблонів:

- pat_Solid: неперервна кисть.
- pat_Horz: горизонтальне штрихування.
- pat_Vert: вертикальне штрихування.
- pat_FDiag: штрихування з нахилом під 45°.

Ось модифікація bkg(W), що видає червоний трикутник:

```
bkg(W) :- Brush= brush(pat_Solid, color_Red),
          P= [pnt(0,0), pnt(10,80), pnt(150, 100)],
          W:setBrush(Brush), W:drawPolygon(P).
```

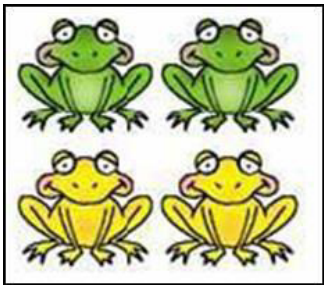
Нарешті, ось версія bkg(W), яка креслить еліпс і очищує в ньому зелений прямокутник.

```
bkg(W) :- R= rct(40, 60, 150, 200),
        W:drawEllipse(R), R1= rct( 60, 90, 140, 130),
        W:clear(R1, color_Green).
```

Останнє, чому ви навчитеся в цьому розділі, це те як додавати .bmp зображення до вашого вікна. От як треба модифікувати метод bkg(W):

```
bkg(W) :- P= vpi::pictLoad("frogs.bmp"),
        W:pictDraw(P, pnt(10, 10), rop_SrcCopy).
```

Перший крок - завантажити зображення з файлу. Це зроблено використанням предиката pictLoad/1. Наступний крок - намалювати зображення. У даному прикладі зображення P розміщене в точці pnt(10, 10).



frogs.bmp

A3mask.bmp

A3.bmp

Часто необхідно намалювати маленьке зображення поверх іншого, прибравши задній фон з маленького зображення. Наприклад, ви бажаєте розмістити орла серед жаб з попереднього прикладу.

Перший крок - намалювати маску орла, використовуючи rop_SrcAnd. Маска показує чорну тінь орла на білому фоні. Наступний крок - намалювати самого орла, використовуючи rop_SrcInvert. Орел повинен абсолютно точно відповідати своєї масці. Програма, що вставляє орла серед жаб, приведена нижче, і реалізована в папці *paintEagle,* у прикладах.

Проект paintEagle у точності відповідає проекту painting, окрім визначення bkw(W).

```
% File dopaint.cl
class dopaint
        open core
predicates
        bkg:(windowGDI).
end class dopaint

%File: dopaint.pro
implement dopaint
        open core, vpiDomains
clauses
        bkg(W) :- P= vpi::pictLoad("frogs.bmp"),
```

```
                W:pictDraw(P, pnt(10, 10), rop_SrcCopy),
                Mask = vpi::pictLoad("a3Mask.bmp"),
                Eagle= vpi::pictLoad("a3.bmp"),
                W:pictDraw(Mask,pnt(50, 50), rop_SrcAnd),
                W:pictDraw(Eagle,pnt(50, 50), rop_SrcInvert).
end implement dopaint
```

14. **Створення елементів управління користувача.** Ми малювали безпосередньо на полотні форми. Проте, доброю ідеєю буде створити призначений для користувача елемент управління, і використовувати його для малювання.

Створіть новий проект

```
Project Name: customcontrol
Object-oriented GUI (pfc/gui)
```

Виберіть ***пункт*** *File/New in New Package* з меню завдань IDE. У діалоговому вікні *Create Project Item*, виберіть елемент *Control* на лівій панелі. Назвою нового елементу управління буде canvas.

```
Name: canvas
New Package
Parent Directory [ ]
```

Залиште поле *Parent Directory* пустим. Натисніть кнопку *Create*.

На екрані з'явиться прототип полотна. Ви можете закрити його, щоб уникнути захаращування IDE. Побудуйте додаток, щоб включити в нього новий елемент управління.

Додайте нову форму до дерева проекту. Виберіть *File/New in New Package* у меню завдань і виберіть пункт *Form* на панелі в лівій частині діалогового вікна *Create Project Items*. Нехай назва нової форми буде greetings. Після ухвалення установок за умовчанням ви отримаєте прототип форми greetings.

Виберіть символ ключа, який з'явиться на панелі *Controls*. Клацніть на поверхні форми greetings. Система покаже вам меню, що містить список нестандартних елементів управління; елемент canvas - перший.

Видалите кнопку *Ok* з форми greetings і замініте її кнопкою hi_ctl, як на мал. 2. Побудуйте додаток.

Відправляйтеся в діалогове вікно *Properties* для кнопки hi_ctl і додайте наступний фрагмент до *ClickResponder*:

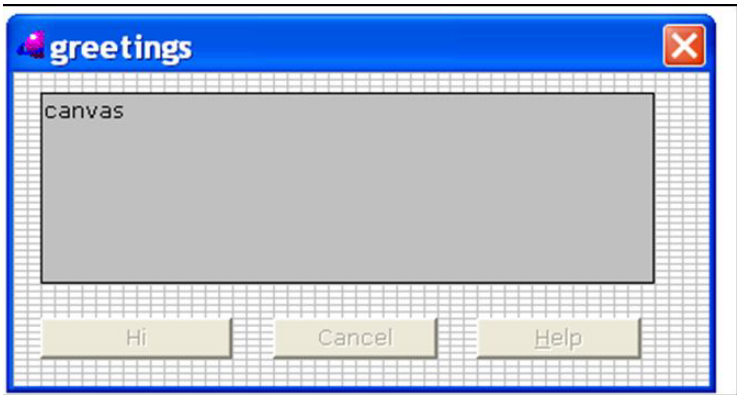
```
clauses
        onHiClick(_Source) = button::defaultAction :-
        W= custom_ctl:getVPIWindow(),
        X= convert(integer, math::random(100)),
        Y= convert(integer, math::random(100)),
        vpi::drawText(W, X , Y, "Hello!").
```

Знову побудуйте додаток. Включите пункт *TaskMenu.mnu→File/New*. Додайте

```
clauses
        onFileNew(S, _MenuTag) :- F= greetings::new(S), F:show().
```

до *TaskWindow.win/Code Expert/Menu/TaskMenu/id_file/id_file_new.*

Побудуйте і запустіть додаток. Виберіть пункт *File/New,* і з'явиться нова форма. Коли ви клацнете в полі *canvas* на новій формі, додаток надрукує теплі вітання.



Мал. 2. Форма з призначеним для користувача елементом управління

1. Основні команди оболонки ЕС "GURU"

BUILD <ім'я набору правил> - використовується для створення, модифікації і компіляції набору правил.

COMPILE <ім'я набору правил> - створює файл набору правил, що відкомпілювалися.

CONSULT <ім'я експертній системи> - звернення до ЕС за консультацією.

WHY <вираз> - пояснює, чому "GURU" використовувала конкретне правило.

HOW <вираз> - пояснює, як "GURU" знайшла значення змінної.

TEXT <ім'я файлу> - запускається текстовий редактор "GURU".

CLEAR - очистити екран.

INPUT <змінна> <USING шаблон><WITH вираз> - введення даних в змінну з використанням шаблону (USING) і з підказкою (WITH).

Шаблони:

a - для буквених символів (латинський шрифт);

c - для букви або числа;

d - для цифри, знаку (+ або -) або десяткової точки;

e - перетворення в символи нижнього регістра;

n - символ шаблону, який сприймає тільки цифри в займаній їм позицій;

r - для символів ASCII;

u - перетворює в символи верхнього регістра.

Наприклад:

INPUT num USING "dddd" with "Введіть номер"

На екрані з'являється текст:

Введіть номер __

Розглянемо ще одну команду "GURU".

OUTPUT <ім'я змінної> <USING шаблон> - виводить на екран змінну або рядок.

Шаблони. В цій команді аналогічні шаблонам для команди INPUT.

Наприклад:

OUTPUT "Лабораторна робота N1"

Виводить на екран:

Лабораторна робота N1

Інший приклад:

OUTPUT num

Виводить на екран значення змінної num. Приведемо також перелік інших команд "GURU":

HELP - виводить довідкову інформацію;

RUN - виконує зовнішню програму;

DIR - проглядає директорію;

BYE - виходить з режиму;

RELEASE - звільняє пам'ять, видаляючи дані і програми "GURU";
PEPFORM - виконує процедуру;
WAIT - припиняє обробку до натиснення будь-якої клавіші.

2. Вирази і функції оболонки ЕС "GURU"

Арифметичні:
+ - складання;
- - віднімання;
* - множення;
/ - ділення;
** - піднесення до степеня;
MOD - ділення по модулю.

Порівняння:
EQ = - дорівнює;
NE <> - не дорівнює;
GT > - більше ніж;
LT < - менше ніж;
GE < = - менше або дорівнює;
LE > = - більше або дорівнює.

Логічні:
NOT - ні;
AND & - і;
OR - або;
XOR - що виключає "або";
= - привласнення;
( ) - індекси масиву.

Строкові:
+ - зчеплення рядків;
' - лапка;
$ - символ відповідності символу;
* - символ відповідності рядка.

Числові функцій:
ABS - абсолютне значення;
ARCSIN - арксинус;
EXP - е в ступені;
INIT - ініціалізує масив;
LEN - визначає довжину рядка;
LN - обчислює натуральний логарифм;
LOG - обчислює логарифм за основою 10;
MAX - найбільше з двох чисел;

MENU - створює меню;
MIN - менше з двох чисел;
RAND - випадкове число;
SIN - синус;
SQRT - квадратний корінь.

Символьні функції:
CHR - перетворює код ASCII в його символьний еквівалент;
VAL - перетворює символ в його код ASCII;
INIT - ініціалізує масив;
SUBSTR - виділяє підрядок з рядка;
TIME - повертає поточний час;
TOSTR - перетворює числа в символи;
TONUM - перетворює рядок в число;
TRIM - відсікає кінцеві пропуски;
TYPE - тип змінної.

Логічні функції:
ALPHASTR - чи весь рядок складається з букв;
INIT - ініціалізує масив.

3. Опис змінних середовища оболонки ЕС "GURU"

№	Ім'я	Тип	Опис	Значення за умовчанням
1	E.BELL	Логічний	Лунає дзвінок, якщо вводиться недійсне значення	TRUE
2	E.DECI	Числовий	Встановлює кількість цифр праворуч від десяткової точки	2
3	E.HRES	Числовий	Встановлює ступінь відповіді на команду HOW від 0 (немає відповіді) до 6 (найбільш докладна відповідь)	4
4	E.LLOG	Числовий	Задає довжину логічного шаблону за умовчанням (максимум - 5)	5
5	E.LSTR	Числовий	Задає довжину символьного шаблону (максимум - 255)	15
6	E.LNUM	Числовий	Задає довжину для числового шаблону (максимум - 14)	14

7	E.OCOH	Логічний	Вивід даних на дисплей	TRUE
8	E.OPRN	Логічний	Вивід всіх вихідних даних на принтер	FALSE
9	E.PDEP	Числовий	Довжина друкарської сторінки	60
10	E.WHN	Символьний	Указує, коли з'являються команди FIND при знаходженні невідомої змінної: N - ніколи; L - тільки як останній засіб; F - перш ніж буде зроблена спроба оцінити значення невідомих змінних	L

4. Графічні засоби оболонки ЕС "GURU"

Мета - знайомство з графічними засобами "GURU", застосуванням їх у прикладних програмах.

За допомогою команд графіки можна створити різні графіки даних з ЕВ, статистики робочих змінних і масивів.

Графіки системи "GURU" можуть створюватися на замовлення за допомогою кольорів, моделей, діапазонів масштабів. Вони можуть зберігатися, друкуватися і викреслюватися. Коли необхідно викреслювати граф, то за умовчанням він займає весь екран.

При необхідності граф може займати вказану частину екрану. При цьому, якщо не вказаний діапазон даних, що виводяться, він розраховується автоматично так, щоб займати максимальний екран.

Управління за допомогою утилітних змінних і змінних середовища

Деякі змінні типу "середовище" і "утиліта" призначено для побудови і оформлення графіків. Перерахуємо основні з них:

Таблиця 1

Ім’я	Значення	Тип	Значення за умовчанням
E.GRID	Фонова сітка для графіків	Лог.	TRUE
E.WFU	Перед віддаленням графічного екрану з дисплея дочекатися, поки користувач не натисне на клавішу.	Лог.	TRUE
#XLABEL	Метка, яка повинна використовуватися на вісі X графіка	Симв.	
#YLABEL	Метка, яка повинна використовуватися на вісі Y графіка	Симв.	

E.BACG визначає фоновий колір графічного екрану;

E.DECI визначає число цифр праворуч від десяткової точки і числах осі і відсотках кругової діаграми;

E.FOHG визначає основний колір графічного екрану;

#TITLE використовується як заголовок графіка.

Крім цих, є певні змінні, які використовуються виключно для графіки. Вони приведені в таблиці 1.

Змінна E.GRID управляє відтворенням фонових сіток для графіків. Сітка складається з горизонтальних і вертикальних ліній.

Якщо E.GRID = TRUE, то сітка з'явиться.

Приведемо перелік команд для роботи з графічними засобами.

Команда PLOT BAR

Дозволяє накреслити гістограму за початковими даними.

PLOT <тип гістограми> BAR FROM <блок> AT <розміщення>

<блок> - блок осередків EB або блок масивів. Якщо <блок> опущений, то мається на увазі блок, що специфікується в попередній команді PLOT.

<тип гістограми> може приймати значення:

SOLID - тривимірна;

COMULATIVE - комулятивна;

STAKED - поверхова (стовпчики в одній групі ставляться один на одного).

<розміщення> може приймати значення:

TOP, TOP LEFT, BOTTOM, TOP RIGHT, LEFT, BOTTOM LEFT, RIGHT, BOTTOM RIGHT.

Якщо AT <розміщення> опущено, то граф займає весь екран.

Приклади:

PLOT BAR FROM #E12 TO #G15

PLOT SOLID BAR FROM AR(1,1) TO AR(4,3) AT TOP PLOT STAKED BAR AT LEFT.

Команда PLOT PIE

Будує кругову діаграму ("пиріг"). Приведемо один з варіантів написання команди: PLOT LABELED PERCENT PIE FROM.

Команда повинна супроводжуватися параметрами:

<джерело даних> AT <розміщення>

<джерело даних> - чисельний вираз, блок осередків або блок масивів EB. Якщо "FROM <джерело даних>" опущено, то використовується <джерело даних> попередньої команди PLOT.

Слово PERCENT в команді - необов'язкове. Якщо воно присутнє, то поряд з сегментами кругової діаграми відтворюються відсотки.

Слово LABELED може використовуватися тільки якщо всі блоки в джерелі даних мають точно дві колонки. Коли ми будуємо кругову діаграму з використанням LABELED, значення відліків в другій колонці кожного блоку використовуються для задання розмірів сегментів. Ці сегменти маркіровані відповідними значеннями з перших колонок (до 6 символів в кожному

позначенні). AT <розміщення> використовується аналогічно команді PLOT BAR.

Приклади:

PLOT PIE FROM #L19 TO #N30 AT BOTTOM

PLOT LABELED PIE FROM #B5 TO #C15

PLOT PERCENT LABELED PIE FROM #A1 TO #B12

PLOT PIE FROM B(1,3) TO (2,7) AT BOTTOM

Команда PLOT LINE

Будує лінійний графік.

PLOT LINE FROM <блок> AT <розміщення>

<блок> - див. опис команди PLOT BAR; AT <розміщення> - див. опис команди PLOT BAR. Лінійний граф має X (горизонтальну) і Y (вертикальну) осі. Окремі точки з'єднуються в графі лініями.

Приклади:

PLOT LINE FROM #G7 TO#MI2 AT LEFT

PLOT LINE FROM ARY(2,3) TO ARY(10,10);

Команда PLOT FUNCTION

Будує функціональні лінії.

PLOT FUNCTION = <вираз> FROM <нижня межа> TO <верхня межа> BY <приріст> AT <розміщення>

<вираз> - функціональна залежність, яка виводитиметься на екран;

<нижня межа>, <верхня межа> - область визначення графіка, що виводиться;

<нижня межа> повинна бути менше <верхньої межі>.

Якщо <вираз> містить змінну X, тоді X є незалежною змінною функції. Під час обробки цієї команди "GURU" ігнорує будь-які раніше існуючі змінні, які мали ім'я X.

BY <приріст> - визначаються додаткові точки X - Y координат;

AT <розміщення> - див. опис команди PLOT BAR.

Приклади:

PLOT FUNCTION = X*X*4+1 FROM 5 TO 2 BY 1

PLOT FUNCTION = 3/SQRT(X) FROM 15.2 TO 18.9 BY 3.5

PLOT FUNCTION = 5*X*X*X BY .5

PLOT FUNCTION = #D5 + X FROM 5 TO 50 BY 10 AT TOP

Команда RANGE

Указує діапазон значень осей координат, використовуваних при побудові графіків і контролює відстань між відмітками на осях.

RANGE ACROSS FROM <нижня межа> TO <верхня межа> BY <приріст>

<нижня межа> <верхня межа> <приріст> - див. опис попередньої команди.

Для графіка, що включає числові дані на осях X, Y, з'являються рівно розміщені і помічені відмітки. Кожна вісь може контролюватися незалежно від інших. Для осі X необхідно використовувати ACROSS, для осі Y - UP.

Команда PATTERN

Для виконання команди необхідно вказати тип заповнення, рядок, відмітку і типи кольорів, використовуваних при викреслюванні графіка.

PATTERN ORDER FOR AREA (LINE, MARK, COLOR) <коди>

<коди> - список одного або декількох чисельних виразів, що вказують порядок, у якому використовуються зразки командою PLOT.

Наприклад, якщо вводиться команда PATTERN ORDER FOR AREA, то <коди> визначають тип заповнення площ:

1 - вертикальні лінії;
2 - горизонтальні, лінії;
3, 4, 6 - діагональні лінії;
5 - вертикальні + горизонтальні;
7 - безперервний.

Якщо вводиться PATTERN ORDER FOR LINE, то визначаються лінії на екрані:

1 - лінія ----------;
2, 3, 4 - лінія --- --- ---;
5 - лінія .....;
6, 7, 8 - лінія ---.---.---.--;

Якщо вводиться PATTERN ORDER COLOR, то коди визначають колір для малювання ліній і закрашення площ: R - червоний; O - жовтий; Z - блакитний; G - зелений; U - синій; H - фіолетовий; W - білий; A - чорний.

Наприклад:

PATTERN ORDER FOR AREA 1, 4, 7, 8

Тоді перша команда PLOT BAR виводить гістограму з типом заповнення 1, друга - 4, третя - 7, четверта - 8.

PATTERN ORDER FOR LINE 1

Всі лінії 1.

PATTERN ORDER FOR COLOR "RGOGRO"

Перший графік буде кольору R, другий G, третій O і т.д.

Команда PLOT TO

Дозволяє запам'ятати графічний екран у файлі.

PLOT TO <ім'я файлу>

Наприклад: PLOT TO "MY_FILE", PLOT TO "MY_GRAPH".

Команда PLOT FROM

Дозволяє завантажити графічний екран з диска.

PLOT FROM <ім'я файлу>

Наприклад:

PLOT FROM "MY_FILE"

PLOT FROM "MY_GRAPH".

6. Приклад використання нечіткої логіки

```
GOAL: whattodo

/*Цей набір правил дозволить вам одержати ряд порад*/
/*на тему "як встигнути на іспит" в залежності від запізнення і*/
/*важливості своєчасного приходу.*/
/*На питання системи слід вводити відповідне*/
/*значення булевої змінної (да - Y, ні -N)*/
/*ну і, звичайно, на прохання системи ввести число – відповідне*/
/*число.*/

INITIAL:
clear
release variable /*прибираємо непотрібні нам змінні*/
e.lstr=250 /*максимальна довжина рядка*/
output "ДЕНЬ ДОБРИЙ, МІСТЕР (MICIC)."
e.cfco="m"
e.cfjo="m"
e.cfva="mm"
output
output "У ВАС СЬОГОДНІ ІСПИТ, А ВИ ПРОКИНУЛИСЯ ДУЖЕ"
output "ПІЗНО... ВАМ, ПРИРОДНО, ТРЕБА ВСТИГНУТИ НА НЬОГО,
АЛЕ ЯК?"
output "МИ ПОСТАРАЄМОСЯ ДАТИ ВАМ ПОРАДУ, ЯК,
ВИХОДЯЧИ З "
output "СИТУАЦІЇ, ЩО СКЛАЛАСЯ, ВАМ СЛІД ПОСТУПИТИ. АЛЕ
ДЛЯ "
output "ЦЬОГО ВИ ПОВИННІ НАДАТИ МЕНІ ІНФОРМАЦІЮ."
output " ОТЖЕ, ПОЧНЕМО ..."
output
lating = "Y"
output "СКАЖІТЬ, ВИ ДІЙСНО СПІЗНЮЄТЕСЯ (Y/N)?"
input lating str using "a"
lating = lating cf 70

DO:
clear
output "ОСЬ ЩО МЕНІ ЗДАЄТЬСЯ ПРИЙНЯТНИМ В ДАНІЙ
СИТУАЦІЇ."
output
output whattodo
RULE: R1
IF: mainexam and biglate
THEN: whattodo=" БЕРІТЬ ТАКСІ НА ВЕСЬ ШЛЯХ ДО ІНСТИТУТУ."
```

whattodo = whattodo + "У ТАКОЙ СИТУАЦІЇ ГРОШІ ЗНАЧЕННЯ НЕ "
whattbdo = whattodo + "МАЮТЬ."
RULE: R2
IF: not mainexam
THEN: whattodo = "ЗАСПОКОЇТЕСЯ, НА НЕ ДУЖЕ ВАЖЛИВИЙ ІСПИТ"
whattodo = whattodo + "НЕ ВАРТО СИЛЬНО ПОСПІШАТИ. ПОВІРТЕ"
whattodo = whattodo+" ВАМ ПРОБАЧАТЬ ВАШЕ ЗАПІЗНЕННЯ АБО НАВІТЬ"
whattodo = whattodo + "ВІДСУТНІСТЬ. ОТЖЕ, БЕЗ СПІШКИ ПО-"
whattodo=whattodo+"ЇДЖАЙТЕ НА АВТОБУСІ." cf 80
RULE: R3
IF: not biglate and mainexam
THEN: whattodo = "НЕ ХВИЛЮЙТЕСЯ, ВСЕ ЩЕ БУДЕ ДОБРЕ. ВАМ"
whattodo = whattodo + " БАЖЕНО УЗЯТИ ТАКСІ НА ЧАСТИНУ ШЛЯХУ"
whattodo = whattodo + "НАПРИКЛАД, ДО ЯКОГО-НЕБУДЬ "
whattodo = whattodo + "ВУЗЛОВОГО ПУНКТУ (ДО МЕТРО, АВТО"
whattodo = whattodo + "СТОЯНКИ). " cf 70
RULE: R4
IF: onlyge4
THEN: mainexam = false cf 85
RULE: R5
IF: veroyatn >= 90
THEN: mainexam = false cf 65
RULE: R6
IF: (veroyatn <90) and not onlyge4
THEN: mainexam = true cf 75
RULE: R7
IF: (lating <> "Y" cf 40) and (lating <> "y" cf 50)
THEN: whattodo = "ВСЕ ГАРАЗД. БАЖАЮ ВАМ НІ ПУХА." cf 70
RULE: R8
IF: howcommon < onwalk + bymetro + bybus+15
THEN: biglate = true cf 60
RULE: R9
IF: howcommon >= onwalk + bymetro + bybus+15
THEN: biglate = false cf 80
VAR: WHATTODO
FIND: whattodo = "ЖАЛКУЮ, Я НЕ ЗНАЮ, ЩО ВАМ ПОРАДИТИ..."
LABEL: порада як діяти в даній ситуації.
VAR: MAINEXAM
LABEL: майбутній іспит - важливий.

VAR: BIGLATE
LABEL: поточне запізнення - значне.

```
VAR: LATING
LABEL: ви реально спізнюєтеся.

VAR: HOWCOMMON
FIND: output
output "СКІЛЬКИ ХВИЛИН ВАМ ДОБИРАТИСЯ ДО ІНСТИТУТУ "
output "ГРОМАДСЬКИМ ТРАНСПОРТОМ?"
input howcommon num using "nnn" cf 70
LABEL: час в дорозі до інституту.

VAR: ONWALK
FIND: output
output "СКІЛЬКИ ХВИЛИН ВАМ ДОВОДИТЬСЯ ЙТИ ПІШКИ?"
input onwalk num using "nn" cf 75
LABEL: час пішого пересування.

VAR: BYMETRO
FIND: output
output "СКІЛЬКИ ХВИЛИН ВАМ ДОВОДИТЬСЯ ПРОВОДИТИ В "
output "МЕТРО?"
input byimetro num using "nnn" cf 60
LABEL: час проїзду в метрополітені.

VAR: BYBUS
FIND: output
output "СКІЛЬКИ ХВИЛИН ВАМ ДОВОДИТЬСЯ ПРОВОДИТИ В "
output "АВТОБУСІ?"
input bybus num using "nnn" cf 50
LABEL: час проїзду в автобусі, тролейбусі, трамваї.

VAR: ONLYGE4
FIND: output
output "НА МАЙБУТНЬОМУ ІСПИТІ НЕ СТАВЛЯТЬ МЕНШЕ 4?"
input onlyge4 logic cf 60
LABEL: на майбутньому іспиті не ставлять менше 4.

VAR: VEROYATN
FIND: output
output "ЯКА ОБ'ЄКТИВНА ВІРОГІДНІСТЬ ОТРИМАННЯ ВАМИ"
output " БАЖАНОЇ ОЦІНКИ?"
input veroyatn num using "nn" cf 60
LABEL: об'єктивна вірогідність отримання вами бажаної оцінки.

END:
```

7. Приклад програми, що виводить дані з електронної відомості

Нехай ми маємо ЕВ, в якій визначається заробітна платня:

	A	B	C
1	Місяць	Номер місяця	Розмір окладу
2	Січень	1	1000
3	Лютий	2	1000
4	Березень	3	1100
5	Квітень	4	1200
6	Травень	5	1300
7	Червень	6	1400

Напишемо програму, яка виводить дані з ЕВ на екран у вигляді графіків.

```
e.lstr = 80 /*ініціалізація змінних*/
e.deci = 2
e.lnum = 8
while true do clear /*очистка екрана*/
vibor = 0
output
output "Виберіть необхідний пункт"
output
output "1 - Побудова гістограми"
output "2 - Побудова кругової діаграми"
output "3 - Побудова графіка"
output "4 - вихід"
input vibor with "Ваш вибір:"
test vibor
   case 1:
      #title = "Гістограма"
      plot bar from #B2 to #C7
      break;
  case 2:
      #title = "Кругова діаграма"
      plot labeled percent pie from #B2 to #C7
      break;
   case 3:
      #title = "Лінійчатий граф"
      plot line from #B2 to #C7
      break;
   case 4:
clear
return
   endtest
endwhile
```

КОНТРОЛЬНІ ПИТАННЯ З ДИСЦИПЛІНИ

Перелік питань до залікових кредитів

Заліковий кредит 1

1. Як визначаються поняття: змінна, константа, функція, терм?
2. Що таке диз’юнкт?
3. Яку ієрархію диз’юнктів Ви можете привести?
4. Як визначається факт, правило й питання?
5. Що таке база знань?
6. Що таке інтерпретація?
7. Яким прикладом можна проілюструвати принцип резолюції?
8. Як визначити поняття підстановки?
9. Що таке уніфікація?
10. Опишіть мовою логіки першого порядку властивості операції додавання, множення.
11. Опишіть мовою логіки першого порядку властивості відносини рівність.
12. Використовуючи формули скорочення запишіть розв’язки попередніх вправ за допомогою зв'язувань "не" й "або".
13. Опишіть мовою Пролог склад своєї родини.
14. Складіть базу знань, що описує країни Європи, Азії, інших материків.
15. Напишіть мовою Пролог таблицю множення чисел від 1 до 10. Яка кількість речень потрібно для запису цієї бази знань?
16. Напишіть мовою Пролог періодичну таблицю хімічних елементів.
17. Опишіть мовою Пролог обчислення площ геометричних фігур: трапеції, трикутника, паралелограма.
18. Опишіть обчислення площі кругу й довжини окружності. Яка точність обчислень цих величин? Чи можна обчислити радіус кола по довжині окружності?
19. Мовою Пролог напишіть базу знань, у якій визначається функція Хевисайда.
20. Які складності можуть виникнути в базі знань про матерів, якщо їх діти будуть тезками?
21. Напишіть програму на Пролозі, що знаходить ім'я матері хлопчика.
22. Написати мовою Пролог базу знань, що описує обчислення факторіала.
23. Написати мовою Пролог базу знань, що описує обчислення суми чисел натурального ряду.
24. Написати мовою Пролог базу знань, що описує обчислення суми квадратів чисел натурального ряду.
25. Які суттєві риси властиві аксіоматичної теорії?
26. Які підходи синтезує логічний підхід до подання знань?
27. Що є призначенням бази знань?

28. Чим пояснюється виникнення формально-логічних методів подання знання?

29. Що є метою застосування числення предикатів?

30. Які методологічні принципи є важливими для моделей подання знань у пам'яті інтелектуальної системи?

31. Чим відрізняється бази знань від баз даних?

32. Що таке клаузи?

33. Як поділяються знання за ступенем їх структурованості?

34. Перерахуйте основні недоліки представлення знання за допомогою аксіоматичних систем.

35. Що необхідно для забезпечення успіху методу резолюцій?

36. Що відображають глобальні стратегії пошуку розв'язків задачі?

37. Які припущення необхідні для побудови аксіоматичних моделей?

38. Яким вимогам повинні відповідати аксіоми?

39. У чому полягає процес побудови аксіоматичної моделі?

40. У чому полягає зміст клауз Хорна?

41. Поясніть, які особливості визначень поняття "інтелектуальна система"?

42. Що відображає мета інтелектуальної системи?

43. Як поділяються цілі на підцілі при розв'язанні логічних задач?

44. Які основні пошукові стратегії для задачі пошуку шляхів на графі?

45. Що таке коректність і повнота систем логічного виводу?

46. Навіщо виконуються підстановки та уніфікації виразів?

47. Що забезпечує процедурна інтерпретація клауз Хорна?

48. Які властивості має структура інтелектуальної системи?

49. Які особливості та характеристики систем штучного інтелекту Ви знаєте?

50. Порівняйте можливості хорновських та нехорновських клауз для представлення знань.

51. Які основні особливості логіки висловлювань?

52. Чим пояснюється широке розповсюдження формально-логічних методів?

53. Які особливості двонаправленого пошуку розв'язків?

54. Що таке продукційна модель подання знання?

55. За якими ознаками класифікуються стратегії пошуку розв'язків?

56. Чи може існувати універсальна модель подання знання?

57. Розкрийте поняття складності логічної програми та основні підходи до її вимірювання.

58. Що таке загальне правило резолюцій?

59. Яка ефективність методів пошуку розв'язків при різних формах подання знань?

60. Які особливості пошуку розв'язків на мові клауз Хорна?

61. У чому суть використання негативних цілей та тверджень у методі резолюцій?

Заліковий кредит 2

1. Що відноситься до простих правил типу FUZZY-FUZZY?
2. Що таке нечітка інформація?
3. Як визначається функція належності для нечітких множин?
4. Визначити значення ступені належності, які відповідають розподілу можливостей терміну «МОЛОДИЙ».
5. Що таке нечіткий факт?
6. Конкретизація, ієрархія та наслідування фреймів.
7. Поняття та властивості продукційної моделі.
8. Основні стратегії вирішення конфліктів у продукційних системах.
9. Які представлення неточних знань Ви знаєте?
10. Які стратегії неточного виводу в умовах невизначеності?
11. Поняття універсума в теорії Заде.
12. Процедури управління системою продукції.
13. Які аксіоми тризначної логіки Лукашевича?
14. Яка структура продукційної моделі?
15. Що таке нечітке логічне виведення?
16. Формування фреймових сценаріїв предметної області.
17. В чому полягає трирівнева архітектура семантичних мереж?
18. Управління онтологіями, створення, пошук і доступ до онтологій.
19. Які способи задавання семантичних мереж?
20. Які властивості фреймів Ви знаєте?
21. Що таке слот?
22. В чому полягає статичність та динамічність фреймів?
23. Приклади команд, які будують гістограму, діаграму, лінію з блоку осередків електронної відомості.
24. Яким чином вводяться команди "GURU" в режиму обробки ЕВ?
25. Що таке визначення і значення осередку ЕВ?
26. Як можна отримати з "GURU" визначення і значення осередків ЕВ?
27. Якими командами завантажується ЕВ з файлу і записується у файл?
28. Назвіть переваги і недоліки командного режиму "GURU".
29. Які команди "GURU" використовуються для логічних переходів за умовами?
30. Якими способами можна консультуватися з набором правил?
31. Які відмінності в застосуванні E.CFJO, S.CFCO і E.CFVA?
32. Що роблять операторів "+=" і "- ="?
33. Навіщо потрібна змінна E.UNKN?
34. Як класифікуються методи видобування знань в експертів?
35. Пасивні та активні методи видобування знань.
36. Групові методи видобування знань.
37. Що є суттю експертних оцінок?
38. Які основні труднощі отримання інформації від експертів?
39. Що таке «когнітивний стиль»?

40. Перерахуйте і назвіть способи, за допомогою яких можна визначити або привласнити значення змінним в "GURU" (у діалоговому і програмному режимах).

41. Формалізування фреймів засобами об’єктно-орієнтованого програмування.

42. Назвіть основні риси відмінності і схожості між мовою "GURU" і якою-небудь поширеною мовою високого рівня.

43. Який алгоритм створення ЕС?

44. Змінні яких типів можна організувати в "GURU"?

45. Що таке прямий і зворотний вивід? Назвіть основні відмінності між ними?

46. Характеристика експертної системи MYCIN.

47. Семантичні мережі, що задають значення.

48. Семантичні графи простого виду.

49. Механізми «пристосування» фрейму до реальної ситуації.

50. Структура фрейму.

51. Логічне виведення за недостовірних знань.

52. Базові елементи фреймів.

53. Фрейми-структури.

54. Переваги ЕС перед людиною-експертом.

55. Особливість фрейм-підходу до проблеми подання знань.

56. Особливості ЕС, що відрізняють їх від звичайних програм

57. З яких частин складається правило ЕС "GURU"?

58. Типи змінних ЕС "GURU" і їх особливості.

59. Скільки інтерфейсів має "GURU"?

60. Що роблять команди INPUT і OUTPUT? Їх особливості.

Перелік питань з курсу

1. Поняття знання.
2. Представлення знання у вигляді продукції.
3. Представлення знання у вигляді семантичної мережі.
4. Поняття інтелектуальних задач.
5. Типи процесів мислення.
6. Поняття штучного інтелекту.
7. Поняття процедурних, декларативних і неявних знань.
8. Поняття експертних знань.
9. Поняття метазнання.
10. Продукційні правила у нормальній формі Бекуса-Наура.
11. Поняття дерева синтаксичного аналізу.
12. Поняття семантичної мережі.
13. Фреймова модель представлення знання.
14. Закони силогістики Аристотеля.
15. Поняття формальної аксіоматичної теорії.
16. Правила виводу (виведення) в аксіоматичній теорії.

17. Алгебра висловлень.
18. Логічні операції (заперечення, кон’юнкція, диз’юнкція, імплікація).
19. Поняття та приклади тавтологій.
20. Аксіомами числення висловлень.
21. Числення предикатів. Теорія першого порядку.
22. Означення формули логіки предикатів.
23. Означення кванторів у логіці предикатів.
24. Поняття формули логіки першого порядку.
25. Сколемська нормальна форма формули числення предикатів.
26. Поняття клаузи Хорна.
27. Алгоритм приведення формули числення предикатів до множини диз’юнктів.
28. Хорновський диз’юнкт.
29. Поняття питання, факту, правила.
30. Поняття голови і тіла клаузи.
31. Подання знання у вигляді клауз Хорна.
32. Поняття підстановка у формули логіки першого порядку.
33. Алгоритм уніфікації формул логіки першого порядку.
34. Поняття резольвенти множини диз’юнктів.
35. Метод резолюцій.
36. Метод SLD-резолюції.
37. Поняття логічної програми.
38. Алгоритм пошуку з поверненням (backtracking).
39. Покрокове виконання логічної програми.
40. Предикати управління пошуком з поверненням.
41. Особливості мови логічного програмування Пролог.
42. Алгоритм логічного виводу.
43. Поняття константи, змінної, структури, предикату у мові Пролог.
44. Структура програми на Visual Prolog.
45. Механізм backtracking у мові Пролог.
46. Методики розробки баз знань мовою Пролог.
47. Особливості побудови бази знань засобами Visual Prolog.
48. Переваги мови Пролог над мовами програмування високого рівня.
49. Приклади побудови бази знань засобами Prolog.
50. Структура програми на мові Пролог.
51. Вбудований предикат ТРАСА.
52. Вбудований предикат !, відсікання (cut).
53. Вбудований предикат АБО.
54. Вбудований предикат ДОДАВАННЯ.
55. Вбудовані предикати МНОЖЕННЯ.
56. Вбудований предикат ВИПАДКОВЕ.
57. Вбудований предикат Запис_в.

58. Вбудований предикат ЛІНІЯ.
59. Вбудований предикат КОЛО.
60. Вбудований предикат ЗЧЕП.
61. Вбудований предикат ВИДАЛЕННЯ.
62. Поняття рекурсивних предикатів і функцій.
63. Приклади рекурсивних програм.
64. Низхідна стратегія виконання рекурсивної програми.
65. Висхідна стратегія виконання рекурсивної програми.
66. Рекурсивна криву Пеано та її реалізація мовою Пролог.
67. Структури даних мови Пролог.
68. Поняття голови і хвоста списку.
69. Операція розщеплювання списку на голову і хвіст.
70. Два способи визначення списків у Пролозі.
71. Форма представлення списків у Пролозі.
72. Шаблон (зразок) списку.
73. Операція склеювання двох списків.
74. Типові процедури обробки списків.
75. Метод наївного сортування списків.
76. Метод сортування списків «бульбашки».
77. Сортування списків методом вставки.
78. Швидке сортування «quick» списків.
79. Обробка списків засобами Visual Prolog
80. Однорідні та неоднорідні семантичні мережі.
81. Концепція семантичної павутини.
82. Формат RDF подання знань.
83. Мова OWL (Web Ontology Language) подання знань.
84. Засоби представлення даних та знань в Семантичній мережі.
85. Мережні продукції.
86. Семантичні мережі, що задають значення.
87. Семантичні графи простого виду.
88. Структура фрейму.
89. Базові елементи фреймів.
90. Визначення фрейму в нотації Бекуса-Наура.
91. Фрейми-структури.
92. Фрейми-ролі.
93. Фрейми-сценарії.
94. Фрейми-ситуації.
95. Мережа фреймів.
96. Глобальна система просторових фреймів.
97. Продукційні моделі бази знань.
98. Продукції типу AW=>BR.
99. Продукції типу AW=>BK.
100. Продукції типу AK=>BW.
101. Продукції типу AK=>BK.
102. Продукції типу AK=>BR.

103. Продукції типу AW=>BW.
104. Продукції типу AR=>BW.
105. Продукції типу AR=>BK.
106. Пошук розв’язків продукційної моделі.
107. Поняття недостовірної та нечіткої інформації.
108. Функція належності та операції над нечіткими множинами.
109. Нечітке логічне виведення.
110. Сфери застосування нечіткої логіки.
111. Перетин двох нечітких множин.
112. Об'єднання двох нечітких множин.
113. Означення нечіткої змінної.
114. Означення лінгвістичної змінної.
115. Поняття експертної системи (ЕС). Сфера застосування ЕС.
116. Структура експертної системи та її основні компоненти.
117. Основні режими роботи експертних систем.
118. Етапи розробки ЕС.
119. Характеристика експертної системи MYCIN.
120. Переваги ЕС перед людиною-експертом.
121. Особливості ЕС, що відрізняють їх від звичайних програм.
122. Визначення складу та моделі представлення знань в ЕС.
123. Рівні представлення і рівні детальності знань в ЕС.
124. Організація знань в робочої пам’яті експертної системи.
125. Подання знань в експертній оболонці EsWin.
126. Поняття метазнання ЕС.
127. Поняття релевантних знань.
128. Основні етапи реалізації системи придбання знань.
129. Метод протокольного аналізу виявлення процедурних знань.
130. Методи пошуку в одному просторі.
131. Методи пошуку в ієрархічних просторах.
132. Методи пошуку при неточних і неповних даних.
133. Засоби представлення знань і стратегії управління статичних експертних систем.
134. Характеристика інструментальної оболонки CLIPS.
135. Можливості оболонки експертної системи GURU.
136. Призначення оболонок експертних систем.
137. Інструментальні комплекси для побудови статичних ЕС.
138. Інструментальні комплекси для створення ЕС реального часу.
139. Інструментальні оболонки експертних систем
140. Сучасні тенденції розвитку штучного інтелекту.
141. Причини успіху систем штучного інтелекту.
142. Представлення знань в оболонці експертної системи G2.
143. Основні компоненти експертної системи реального часу.
144. Особливості архітектури експертної системи реального часу.
145. Експертні системи реального часу як важливий напрям розвитку систем штучного інтелекту.

ЗАВДАННЯ ДО САМОСТІЙНОЇ ТА ІНДИВІДУАЛЬНОЇ РОБОТИ

1. Завдання до самостійної роботи

1. Довести, що формула логіки предикатів

$$\exists x(P(x)\to Q(x))\to(\forall x(P(x)\to \exists xQ(x))$$

є тотожно істинною. Чи буде такою обернена до неї імплікація?

2. Довести, що формули логіки предикатів

$$(\forall xP(x)\to \exists xQ(x)) \sim \exists x(P(x)\to Q(x))$$

$$(\exists xP(x)\to \forall xQ(x))\to \forall x(P(x)\to Q(x))$$

$$\exists x(P(x)\wedge Q(x))\wedge \forall x(Q(x)\to\neg R(x))\to \exists x(P(x)\wedge\neg R(x))$$

є логічно загальнозначущими.

3. Показати, що формула логіки предикатів

$$\forall x(P(x)\to Q(x))\wedge \forall x(P(x)\to R(x))\to \exists x(Q(x)\wedge R(x))$$

не є тотожно істинною.

4. Показати, що формула логіки предикатів

$$\forall x(P(x)\to\neg Q(x))\wedge \forall x(R(x)\to P(x))\to \exists x\neg(Q(x)\wedge R(x))$$

є тотожно істинною.

5. Оцінити істинність математичних тверджень:

а) $\forall x\forall y\exists z(x<z\wedge z<y\vee y<z\wedge z<x)$,

б) $\forall x\forall y\exists z(x<y\to x<z\wedge z<y)$

у випадку, коли універсальною множиною є: 1) множина всіх натуральних чисел; 2) множина всіх цілих чисел; 3) множина всіх раціональних чисел; 4) множина всіх дійсних чисел.

6. Довести, що формула логіки предикатів:

$$\forall xP(x)\vee \forall xQ(x) \to \forall x(P(x)\vee Q(x))$$

є тотожно істинною, а обернена до неї імплікація - ні.

7. Показати, що формула логіки предикатів

$$\forall x(P(x)\to Q(x))\to(\forall xP(x)\to \forall xQ(x))$$

є логічно загальнозначущою, а обернена до неї імплікація - ні.

8. Показати, що формула логіки предикатів

$$(\exists xP(x)\to \exists xQ(x))\to \exists x(P(x)\to Q(x))$$

є тотожно істинною, тоді як обернена до неї імплікація не буде такою.

9. Чи є тотожно істинною формула логіки предикатів

$$\forall x(P(x) \sim Q(x)) \sim (\forall xP(x) \sim \forall xQ(x))?$$

$$\exists x(P(x) \sim Q(x)) \sim (\exists xP(x) \sim \exists xQ(x))?$$
$$\forall x(P(x) \to Q(x)) \wedge \forall x(P(x) \to R(x)) \to \exists x(Q(x) \to R(x))?$$

Відповіді обґрунтувати.

10. Довести, що формула логіки предикатів

$$(\forall xP(x) \sim \forall xQ(x)) \to \forall x(P(x) \sim Q(x))$$

не є тотожно істинною, тоді як обернена імплікація до неї тотожно істинна.

11. Вказати вільні і зв'язані входження змінних у запропонованих виразах. Для кожного зв'язаного входження змінної зазначити, яким саме квантором вона зв'язана:

1) $P(x) \to \forall y(Q(y) \vee \exists zP(z) \sim \neg P(y))$;
2) $\exists x(P(y) \to P(x) \wedge \exists z(Q(z) \sim \exists y(Q(y) \vee Q(x))) \wedge \forall x(P(x) \to \neg(\exists yQ(y)))$;
3) $\forall x(x2y>0 \to y>0)$;
4) $x>3 \wedge \forall x \forall y(xy2>0)$;
5) $\forall x \forall y(x<y \to \exists z(x<z \wedge z<y)$;
6) $xy<0 \to \exists z(xyz>0)$.

12. Виразити твердження «*a* є простим числом» через символи операцій логіки предикатів, індивідуального предиката рівності і арифметичні константи «1» та «*» (знак множення).

13. Застосовуючи символіку логіки предикатів, записати умову обмеженості знизу і необмеженості зверху множини всіх натуральних чисел.

14. Універсальна множина - множина всіх натуральних чисел - 1, 2, 3, ..., *n*, ... Записати логіко-математичною символікою твердження: «17 є найменшим числом, яке при діленні на 11 дає в остачі 6». Як зміниться значення істинності твердження, якщо за множину натуральних чисел прийняти {0,1,2,...}?

15. Побудувати інтерпретацію формули логіки предикатів $\forall x \exists yF(x,y) \wedge \forall x \neg F(x,x) \wedge \forall x \forall y \forall z(F(x,y) \wedge F(y,z) \to F(x,z))$ над множиною $M = \{1,2,3\}$, заміщуючи предикатну змінну $F(x,y)$ на $x<y$. Оцінити істинність здобутого при цьому висловлення.

16. Модифікувати наведену нижче ЕС, передбачивши такі ситуації:

а) несправність клавіатури комп'ютера;
б) несправність маніпулятора типу "миша".

goal: computer
/*Цей набір правил дозволить вам одержати ряд порад*/
/*як діяти, якщо раптом ваш комп'ютер при включенні*/

```
/*його в мережу поводиться не так, як звично. І ось, в залежності*/
/*від зовнішнього прояву цих дивностей вам буде*/
/*дана порада, як діяти.*/
/*на питання системи слід вводити відповідне*/
/*значення булевої змінної ( так - true, ні-false)*/
initial:
clear
release variable /*прибираємо непотрібні нам змінні*/
e.lstr=250/*максимальна довжина рядка*/
output "День добрий, містер (місіс)."
output
output "При включенні комп'ютера ви не одержуєте звичайного"
output "результату. Ми постараємося дати вам пораду в цій не-"
output "простій справі. Але для цього ви повинні надати"
output "мені інформацію. Отже, почнемо"
power=true
output "чи спалахує індикатор живлення на вашому комп'ютері?"
input power logic

do:
clear
output "ось що я вам скажу, люб'язний."
output
output computer

rule: r1
if: power
then: output
output"ну хоч живлення в порядку, і те добре!"
output "справний і чи правильно підключений дисплей?"
input displ logic
reason: якщо в порядку живлення, то чи в порядку дисплей.
comment: для того, щоб щось побачити, необхідний пристрій відображення інформації.

rule: r2
if: not power
then: output
computer="перевірьте напругу в мережі. включите живлення"
computer=computer+"або полагодите блок живлення і"
computer=computer+"спробуйте ще раз."
reason: якщо немає живлення, то необхідно його забезпечити
comment: без нормального живлення ніхто вам працювати не буде.
rule: r3
if: not displ
```

```
then: output
computer=" перевірьте напругу. включить дисплей"
computer=computer+"або влаштуєте так, щоб він працював"
computer=computer+"і спробуйте ще раз."
reason: не працює дисплеї. необхідно, щоб він працював.
comment: без працюючого дисплея - не життя.

rule: r4
if: displ
then: output "чи є у вашого комп'ютера жорсткий диск?"

input harddisk logic
reason: якщо живлення комп'ютера і дисплеї в порядку, то треба знати,
чи підключений до вашої машини жорсткий диск.
comment: чи підключений жорсткий диск.

rule: r5
if: harddisk
then: output "відбувається звернення до диска (індикатор "
output "горить)?"
input hdtest logic
reason: якщо встановлений "вінчестер", то перевіримо підключення.
comment: якщо "гвинт"на місці, то чи помічає його система.

rule: r6
if: not harddisk
then: output "встановлені на вашій машині дисководи гничких"
output "дисків (1 і більш)? "
input diskete logic
reason: якщо немає "гвинта", то дисководи на машині встановлені?
comment: чи є дисководи.

rule: r7
if: not hdtest
output "чи встановлені на вашій машині дисководи"

output "гнучких дисків (1 і більш)?
then: input diskete logic
reason: чи встановлені дисководи на машині?
comment: чи є дисководи.

rule: r8
if: hdtest
then: output "чи видається повідомлення про помилку читання?"
input hderror logic
```

reason: чи без помилок проходить операція тестування.
comment: наявність помилок тестування диска.

rule: r9
if: hderror
then: output "чи встановлені на вашій машині дисководи гнучких"
output "дисків (1 і більш)?"
input diskete logic
reason: якщо є помилки в роботі "гвинта", то цікаво, чи встановлені дисководи на машині?
comment: чи є дисководи.

rule: r10
if: diskete
then: computer="Вам слід вставити в дисковод системну "
computer=computer+"дискету відповідного формату "
computer=computer+"і перезавантажити комп'ютер."
reason: якщо "гвинт"не в порядку, то слід використовувати гнучкий диск для завантаження системи.
comment: слід використовувати гнучкий диск для завантаження системи.

rule: r11
if: not diskete
then: computer="Вам краще всього далі не заглиблюватися в "
computer=computer+"цю проблему, а викликати майстра."
reason: якщо не працюють (відсутні) диск і дисководи, то не варто намагатися перезавантажити машину.
comment: не працюють диск і дисководи, отже, не варто намагатися перезавантажити машину.

rule: r12
if: not hderror
then: computer="якщо не з'являється запрошення операційної"
computer=computer+"системи, то перевірте зміст"
computer=computer+"файлів autoexec.bat і config.sys,"
computer=computer+'завантажтесь з дискети. і якщо там "
computer=computer+"все гаразд, перезапишіть ос."
reason: якщо немає помилки читання "вінчестера", то все ще можна виправити, лише б дисковод гнучких дисків був справний.
comment: немає помилки читання "гвинта".

var: power
label: спрацьовування світлової індикації при включенні комп'ютера
end:

17. Модифікувати наведену нижче ЕС, передбачивши такі ситуації:
    а) поява міхурів на забарвленій поверхні;
    б) відшарування фарби після висихання.

```
goal: ways

/*цей набір правил дозволить вам одержати ряд порад по*/
/*усуненню деяких дефектів водних фарб і повідомить*/
/*причини їх появи.*/
/*на питання систем слід вводити*/
/*значення булевої змінної. (так-true, ні-false )*/

initial:
clear
release variable /*прибираємо непотрібні нам змінні*/
e.lstr=250/*максимальна довжина рядка*/
output "день добрий, містер (місіс)."
output
output "після забарвлення поверхні водними фарбами можуть"
output "виявитися дефекти."
output "ми постараємося дати вам пораду з їх усунення і"
output "причинам появи."
output "але для цього ви повинні надати мені всю "
output "інформацію."
output "отже, почнемо ..."
defects=true
output "чи з'явилися дефекти на забарвленій поверхні?"
input defects logic

do:
clear
output "ось що я вам скажу, люб'язний."
output
output reasons
output
output "а ось що вам слід зробити в даній ситуацій."
output
output ways

rule: rl
if: defects
then: output
input circle logic with "ці дефекти - плями?"
reason: якщо є дефекти, то швидше за все це плями.
comment: чи є дефекти плямами.
```

rule: r2
if: not defects
then: output
reasons="немає дефектів на поверхні, це означає що ви "
reasons=reasons+"все зробили правильно."
ways="підіть краще відпочиньте. адже у вас"
ways=ways+"все гаразд."
reason: якщо немає дефектів, то нічого робити не потрібно.
comment: немає дефектів - відпочивай.

rule: r3
if: not circle
then: output
input lines logic with "ці дефекти - смуги? "
reason: якщо дефекти не плями, то швидше за все це смуги.
comment: чи є дефекти смугами.

rule: r4
if: circle
then: output
input fatcrcl logic with "це жирні плями ?"
reason: якщо дефекти плями, то швидше за все це жирні плями.
comment: чи є плями жирними.

rule: r5
if: fatcrcl
then: output
output "забарвлена поверхня є "
output "штукатуркою?"
input sfcshtkr logic
output
reason: якщо плями жирні, то важливо знати, яка поверхня забарвлена.
comment: на штукатурці жирні плями.

rule: r6
if: not sfcshtkr
then: output
output "забарвлена поверхня є"
output "залізобетонною?"
input sfcsteel logic
reason: якщо п'ята жирні, то важливо знати, яка поверхня забарвлена.
comment: на залізобетоні жирні плями.

rule: r7

if: not sfcsteel
them: output
reasons="причина появи цих жирних плям"
reasons=reasons+"мені невідома."
ways="спробуйте звернутися до досвідченішому"
ways=ways+"фахівця."
reason: якщо жирні плями з'явилися на якійсь іншій поверхні, то нічого путнього порадити вам я не можу.
comment: жирні плями на не зарезервованій поверхні.

rule: e8
if: sfcsteel
then: reasons="на залізобетоні сліди невисихаючих "
reasons=reasons+"масел від мастила форм."
ways="очистить поверхню від шару фарби разом з"
ways=ways+"шпаклівкою, промити 5%-м розчином "
ways=ways+"тринатрійфосфату або "
ways=ways+"соди, нейтралізувати поверхню 5%-м"
ways=ways+"розчином соляної кислоти і знов пофарбувати."
reason: якщо жирні плями на залізобетоні, то ця проблема цілком вирішувана.
comment: спосіб усунення жирних плям на залізобетоні.

rule: r9
if: sfcshtkr
then: reasons="на штукатурці залишилися плями невисихаючих"
reasons=reasons+"мінеральних і тваринних масел."
ways="вирубіть штукатурку на ділянці плями, знову "
ways=ways+"відштукатурте і пофарбуйте, промийте поверхню"
ways=ways+"водою і знов пофарбуйте."
reason: якщо жирні плями на штукатурці, то ця проблема цілком вирішувана.
comment: спосіб усунення жирних плям на штукатурці.

rule: r10
if: not fatcrcl
then: output
output "це жовті іржаві плями?"
input rsvcrcl logic
reason: якщо дефекти не жирні плями, то швидше за все це іржаві плями.
comment: чи є плями іржавими.

rule: r11
if: not rsvcrcl

then: reasons="причина появи цих плям мені"
reasons=reasons+"невідома."
ways="попробуйте звернутися до досвідченішого"
ways=ways+"фахівця."
reason: якщо з'явилися якісь інші плями, то нічого путнього порадити вам я не можу.
somment: якісь незарезервовані плями на поверхні.

rule: r12
if: rsvcrcl
tnen: reasons="відбувається теча смолянистих речовин "
reasons=reasons+"через штукатурку і фарбу."
ways="віддалить старий набіл, промити теплим 3% розчином"
ways=ways+"соляної кислоти і, якщо плями невеликі"
ways=ways+", загрунтуйте мідною ґрунтовкою"
ways=ways+"без крейди, а при великих розмірах – щелочним"
ways=ways+", спиртним або каніфольним лаком."
reason: якщо жирні плями на штукатурці, то ця проблема цілком вирішувана.
comment: спосіб усунення жирних плям на штукатурці.

rule: r13
if: lines
then: reasons="недостаточно перемішані пігменти в кольорі"
reasons=reasons+"на забарвленій поверхні?"
ways="промити поверхня і пофарбувати з "
ways=ways+"краско пульта."
reason: якщо смуги на забарвленій поверхні, то ця проблема цілком вирішувана.
comment: спосіб усунення смуг.

rule: r14
if: not lines
then: reasons="причина появи інших дефектів мені"
reasons=reasons+"невідома."
ways="попробуйте звернутися до досвідченішому"
ways=ways+"специалисту."
reason: якщо з'явилися якісь інші дефекти, то нічого путнього порадити вам я не можу.
comment: якісь незарезервовані дефекти на поверхні.

var: defects
label: наявність дефектів на забарвленій поверхні.

end:

18. Модифікувати наведену нижче ЕС, передбачивши такі ситуації:
а) пограбування транспорту, що перевозить банківські цінності;
б) пограбування банку в умовах стихійного лиха.

```
goal: breaking

/*цей набір правил дозволить вам одержати ряд порад по*/
/*організації пограбування банку. на питання системи слід вводити*/
/*відповідне значення булевої зміною*/
/*( так-true, ні-false )*/

initial:
clear
release variable /*прибираємо непотрібні нам змінні*/
e.lstr=250/*максимальна довжина рядка*/
output"день добрий, містер (місіс)."
output
output "всі хочуть пограбувати банк, та не всі можуть..."
output "ми постараємося дати вам пораду в цій непростій справі."
output "але для цього ви повинні надати мені всю"
output "інформацію."
output "отже, поїхали ..."
output
at 17,10 output "скільки у вас вірних людей?"
input people num using "nn"
at 19,10 output "скільки у вас 'стволів' ?"
input guns num using "nn"
at 21,10 output "а скільки ви хочете узяти?"
input nmoney num
output
clear

do:
clear
output
output "ось що я вам скажу, люб'язний."
output
output breaking

rule: r1
if: nmoney >=10000000
then: output
output "а у вас губа не дурка!"
output "а скільки грошей зберігається в банку "
output "готівкою?"
```

input emoney num

reason: якщо ви так багато хочете узяти, то слід знати, чи має в своєму розпорядженні банк таку суму.

comment: ви так багато хочете узяти, отже. слід знати, чи має в своєму розпорядженні банк таку суму.

rule: r2
if: nmoney<10000000
then: output "а скільки грошей зберігається в банку готівкою?"
input emoney num

reason: якщо ви хочете пограбувати банк, то слід знати, чи має він таку суму.

comment: ви хочете узяти чужі гроші, отже, слід знати, чи має в своєму розпорядженні банк таку суму.

rule: r3
if: nmoney<emoney
then: output
output "і так, ви зважилися на справу, а як йдуть"
output "діла з охороною?"
output
output "чи охороняється банк вдень?"

input dayctrl logic
output
output "чи охороняється банк поліцейськими вночі?"

input nitectrl logic

reason: якщо в банку достатньо грошей, то починаємо збирати інформацію про охорону.

comment: банк має достатньо засобів для задоволення ваших потреб. дізнаємося, чи охороняється він вдень і вночі.

rule: r4
if: dayctrl
then: output
output "скільки чоловік є на охороні банку вдень?"

input daycol num using "nn"
output
output "чи є видимим приміщення банку"

output "відеокамерою?"
input kamera logic

reason: якщо вдень банк охороняється, то треба дізнатися про к-ть охорони і одержати інформацію про наявність в приміщенні банку відеокамери.

comment: якщо біля входу вдень є поліцейські, то треба дізнатися скільки їх, і чи є видимим приміщення відеокамерою

rule: r5
if: nitectrl
then: output
output"скількі чоловік є на охороні банку вночі?"

input nitecol num using "nn"
output
output "чи можете ви відключити сигналізацію"

output "щоб вона не спрацювала?"
input signal logic

reason: якщо вночі банк охороняється, то треба дізнатися про к-ть охоронців і чи можна відключити, сигналізацію.

comment: якщо вночі банк охороняють поліцейські, то треба дізнатися скільки їх і чи можна відключити сигналізацію.

rule: r6
if: not nitectrl
then: output
output "чи можете ви відключити сигналізацію"

output "щоб вона не спрацювала?"
input signal logic

reason: якщо вночі банк не охороняється, то треба дізнатися, чи можливо відключити сигналізацію.

comment: якщо вночі банк не охороняють поліцейські, то чи можливо відключити сигналізацію.

rule: r7
if: not nitectrl and signal
then: breaking="операцию треба починати в 2 години 17 хвилин"
breaking=breaking+"за місцевим часом."

reason: якщо банк вночі під поганою охороною, то пограбування пройде цілком успішно. головне почати його в 2 години 17 хвилин. у ніч на п'ятницю.

comment: такі банки треба брати вночі.

rule: r8
if: not nitectrl and not signal and dayctrl

then: breaking="операцию треба починати в 2 години 17 хвилин "
breaking=breaking+"за місцевим часом. але будьте "
breaking=breaking+"обережні, Вам треба все зробити"
breaking=breaking+"за 15 хвилин. тренуйтеся!"
reason: якщо банк вночі під поганою охороною, то пограбування пройде цілком успішно. головне почати його в 2 години 17 хвилин. у ніч на п'ятницю. і врахуйте, у вас всього 15 хвилин.
comment: такі банки треба брати вночі.

rule: r9
if: (nitectrl or not signal and not nitectrl) and not dayctrl and not kamera
then: breaking="такий банк треба брати в 16 годинників 45 хвилин"
breaking=breaking+"удвох, маючи при собі пістолети."
reason: якщо банк вночі під охороною або сигналізацією, але вдень кинутий на свавілля долі, то пограбування пройде цілком успішно. головне почати його в 16 годинників 45 хвилин у п'ятницю.
comment: такі банки треба брати вдень при всьому чесному народі.

rule: r10
if: (nitectrl or not signal and not nitectrl) and not dayctrl and kamera
then: breaking="такий банк треба брати в 16 годинників 45 хвилин"
breaking=breaking+"удвох, маючи при собі пістолети"
breaking=breaking+"і маски."
reason: якщо банк вночі під охороною або сигналізацією, але вдень кинутий на свавілля долі, то пограбування пройде цілком успішно. головне почати його в 16 годинників 45 хвилин у п'ятницю.
comment: такі банки треба брати вдень при всьому чесному народі.

rule: r11
if: dayctrl and nitectrl and (people<nitecol or guns<daycol)
then: breaking="сидели б ви краще удома і не рипалися "
breaking=breaking+"понапрасну."

rule: r12
if: (people or guns)<l and not signal
theh: breaking="сидели б ви краще удома і не рипалися "
breaking=breaking+"по напрасну. при такому розкладі "
breaking=breaking+"ви обов'язково попадетеся. моя "
breaking=breaking+"Вам порада: шукайте людей і "
breaking=breaking+"збирайте зброю."

rule: r13
if: not dayctrl and people>1 and guns>l and nitectrl
then: breaking="у вас непогані шанси на успіх при роботі"
breaking=breaking+"вдень. і якщо ви захопите два "

```
breaking=breaking+"самих спритних хлоп'ят, то при "
breaking=breaking+"непоганому розкладі ви зробите це"
breaking=breaking+"успішно. так, і не забудьте маски"
breaking=breaking+"там напевно встановлена "
breaking=breaking+"відеокамера."
comment: вдень банк не охороняється.

rule: r14
if: daycol<min(people,guns)-2 and nitecol>daycol+1
then: breaking="краще всього брати цей банк штурмом"
breakine=breaking+"так у вас цілком достатньо"
breaking=breaking+"людей. але чи варто робити це?"
comment: вдень контроль слабкіше, ніж вночі.


var: guns
label: к-ть наявних у вашому розпорядженні од. вогнепальної зброї.

var: nmoney
label: планована сума пограбування.

var: people
label: к-ть членів вашої тусовки.

end:
```

19. Модифікувати наведену нижче ЕС, передбачивши такі ситуації:
    а) наявність в холодильнику сира;
    б) наявність в холодильнику пива.

```
goal: dinner

/*цей набір правил дозволить вам. одержати ряд порад по*/
/*приготуванню обіду на бажане число персон в*/
/*залежності від набору і кількості продуктів*/
/*наявних у вашому розпорядженні.*/
 /*на питання системи слід вводити відповідне*/
/*значення булевої змінної ( да-y, ні-n )*/

initial:
clear
release variable /*прибираємо непотрібні нам змінні*/
e.lstr=250/*максимальна довжина рядка*/
output "день добрий, містер (місіс)."
output "пошаривши в своєму холодильнику і виявивши там "
```

```
output "деякі запаси, ви вирішили що-небудь приготувати. але "
output "що саме ви ще не придумали. ми постараємося дати "
output "вам поради, як з наявних продуктів зробити щось"
output "їстівне. але для цього ви повинні надати мені "
output "всю інформацію."
output "отже, почнемо ..."
meatexst="y"
output "скажіть, ви знайшли в холодильнику м'ясо (y/n) ?"
input meatexst str using "a"

do:
clear
output "ось що мені здається прийнятним в даній ситуації."
output
output dinner

rule: rl
if: meatexst<>"y"or meatexst<>"y"
then: output "даруйте, а ви вегетаріанець (y/n)?"
input vegitar str using "a"

rule: r2
if: (meatexst<>"y"or meatexst<>"y") and (vegitar<>"y"or vegitar<>"y")
then: dinner="краще всього візьміть свій товстий гаманець"
dinner=dinner+"і відправляйтеся в ресторан, бо без "
dinner=dinner+"м’яса нормального обіду не вийде."

rule: r3
if: vegitar="y"or vegitar="y"
then: output
output "ну, раз ви вегетаріанець, то, напевно, овочів "
output "у вас навалом (y/n)?"
input vagitables str using "a"

rule: r4
if: meatexst="y"or meatexst="y"
then: output
output "скільки грамів м'яса ви маєте?"
input howmeat num using "nnnn"
output
output "на скільки персон задуманий ваш обід?"
input howfamily num using "n"

rule: r5
if: meatexst<>"y"or meatexst<>"y"
```

```
then: output
output "даруйте, а ви вегетаріанець (y/n)?"
input vegitar str using "a"

rule: r6
if: howmeat/howfamily>300
then: dinner="краще всього візьміть цей шматок м'яса"
dinner=dinner+"наріжте його уздовж волокон на "
dinner=dinner+"не товсті пласти, відбійте, нашпигуйте"
dinner=dinner+"і на 30 хв. відкладіть. потім беріть"
dinner=dinner+"сковороду і смажте на повільному "
dinner=dinner+"вогні. як мовиться, сіль, ананаси"
dinner=dinner+"фісташки – за смаком."

rule: r7
if: (howmeat/howfamily>80) and (howmeat/howfamily<=300)
then: enoughmeat=true

rule: r8
if: enoughmeat
then: output
output "м'ясо - це запорука успіху ."
output
output "чи є у вас капуста (y/n)?"
input kapusta str using "a"
output
input svekla str using "a"with "буряк (y/n) ?".
output
input potetou str using "a"with "картопля (y/n)?"

rule: r9
if: (kapusta="y"or kapusta="y") and (svekla="y"or svekla="y") and
(potetou= "y"or potetou="y")
then: dinner="варить український борщ."

rule: r10
if: howmeat/howfamily<=80
then: enoughmeat=false

rule: r11
if: not enoughmeat
then: dinner="м'яса дуже мало, щоб приготувати перше або"
dinner=dinner+"друге, але мабуть, він вистачить на "
dinner=dinner+"м’ясний салат."
```

```
rule: r12
if: vagitables="y"or vagitables="y"
then: dinner="звалить все в одну купу, потім дрібно покришите"
diriner=dinner+"(можна пропустити через м'ясорубку)"
dinner=dinner+"перекладіть масу, що вийшла, в "
dinner=dinner+"салатниці і подавайте до столу. якщо вам "
dirmer=dinner+"повезе, то всі будуть задоволені. якщо немає -"
dinner=dinner+"не осудіть."


var: dinner
find: dinner="причина появи цих неполадок нам невідома."
label: як поступити, якщо потрібно приготувати обід.

var: meatexst
label: наявність м'яса у вашому холодильнику.

var: enoughmeat
label: чи достатньо м'яса для одного з перших блюд.

var: howmeat
label: приблизна вага шматка м'яса.

var: howfamily
label: кількість персон, запрошених на обід.

var: kapusta
label: чи є в будинку капуста.

var: svekla
label: чи є в будинку буряк.

var: potetou
label: чи є в будинку картопля.

var: vegitar
label: чи є ви вегетаріанцем.
end:
```

20. Модифікувати наведену нижче ЕС, передбачивши такі ситуації:

а) своє погане самопочуття (хвороба);

б) можливість подзвонити екзаменатору в аудиторію, де проходить іспит (попередити).

```
goal: whattodo
```

```
/*цей набір правил дозволить вам одержати ряд порад*/
/*тему "як встигнути на іспит", в залежності від запізнення і/
/*важливості своєчасного приходу.*/
/*значення булевої змінної ( да-y, нет-n )*/
/*ну, і звичайно, на прохання системи ввести число.*/

initial:
clear
release variable /*прибираємо непотрібні нам змінні*/
e.lstr=250/*максимальна довжина рядка*/
output "день добрий, містер (місіс)."
output
output "у вас сьогодні іспит, а ви прокинулися дуже "
output "пізно... вам, природно, треба встигнути на нього, але "
output "як? ми постараємося дати вам раду, як, виходячи з "
output ситуації, що склалася, вам слід поступити. але для "
output "цього ви повинні надати мені всю інформацію."
output "отже, почнемо ..."
output
lating="y"
output"скажіть, ви дійсно спізнюєтеся (y/n)?"
input lating str using "a"

do:
clear
output "ось що мені здається прийнятним в даній ситуації."
output
output whattodo

rule: rl
if: mainexam and biglate
then: whattodo="беріть таксі на весь шлях до інституту. y"
whattodo=whattodo+"такій ситуації гроші значення не "
whattbdo=whattodo+"мають."

rule: r2
if: not mainexam
then: whattodo="заспокоїтеся, на не дуже важливий іспит "
whattodo=whattodo+"не варто сильно поспішати. повірте"
whattodo=whattodo+" Вам пробачать ваше запізнення або навіть"
whattodo=whattodo+"відсутність. отже їдьте"
whattodo=whattodo+"на громадському транспорті."

rule: r3
```

if: not biglate and mainexam
then: whattodo="не хвилюйтеся, все ще буде добре. вам "
whattodo=whattodo+"варто узяти таксі на частину шляху"
whattodo=whattodo+"наприклад, до якого-небудь "
whattodo=whattodo+"вузлового пункту (до метро, авто"
whattodo=whattodo+"стоянки)."

rule: r4
if: onlyge4
then: mainexam=false

rule: r5
if: veroyatn >= 90
then: mainexam=false

rule: r6
if: (veroyatn <90) and not onlyge4
then: mainexam=true

rule: r7
if: lating<>"y"and lating<>"y"
then: whattodo="все гаразд. бажаю вам успіхів."

rule: r8
if: howcommon<onwalk+bymetro+bybus+15
then: biglate=true

rule: r9
if: howcommon >= onwalk+bymetro+bybus+15
then: biglate=false

var: whattodo
find: whattodo="жалкую, я не знаю, що вам порадити..."
label: порада як діяти в даній ситуації.

var: mainexam
label: майбутній іспит - важливий.

var: biglate
label: поточне запізнення - значне.

var: lating
label: ви реально спізнюєтеся.

var: howcommon

```
find: output
output "скільки хвилин вам добиратися до інституту "
output "громадським транспортом?"
input howcommon num using "nnn"
label: час в дорозі до інституту.

var: onwalk
find: output
output "скільки хвилин вам доводиться йти пішки?"
input onwalk num using "nn"
label: час пішого пересування.

var: bymetro
find: output
output "скільки хвилин вам доводиться проводити в "
output "метро?"
input byinetro num using "nnn"
label: час проїзду в метрополітені.

var: bybus
find: output
output "скільки хвилин вам доводиться проводити в "
output "автобусі?"
input bybus num using "nnn"
label: час проїзду в автобусі, тролейбусі, трамваї.
var: onlyge4
find: output
output "на майбутньому іспиті не ставлять менше 4?"
input onlyge4 logic
label: на майбутньому іспиті не ставлять менше 4.

var: veroyatn
find: output
output "яка об'єктивна вірогідність отримання вами"
output "бажаної оцінки?"
input veroyatn num using "nn"
label: об'єктивна вірогідність отримання вами бажаної оцінки.
end:
```

21. Написати програму "GURU", яка будує гістограму, діаграму і графік з електронної відомості.

22. Скласти програму на мові Пролог обчислення добуток елементів заданого числового списку.

23. Скласти програму на мові Пролог обчислення середнього геометричного елементів заданого числового списку.

24. Скласти програму на мові Пролог, що породжує список, що містить натуральні числа від 1, 3, 5, …, 2*N+1 (за збільшенням).

25. Скласти програму на мові Пролог, що породжує список, що містить куби натуральні чисел від 1 до N (за збільшенням).

26. Скласти програму на мові Пролог, що дозволяє видалити із заданого списку натуральних чисел всі парні числа.

27. Складіть програму Пролозі, яка

а) підраховує число людей з однаковими іменами - тезків серед ваших родичів.

б) визначить ім'я наймолодшої і найстарішої людини серед вашої рідні.

28. Записати за допомогою мови ПРОЛОГ наступні правила:

- Якщо тварина має волосся, то це тварина ссавець;
- Якщо тварина має пір'я, то це тварина птах;
- Якщо тварина може літати і відкладає яйця, то це тварина птах;
- Якщо тварину їсть м'ясо, то це тварина хижак;
- Якщо тварина має гострі зуби і тварину має кігті і його очі дивляться вперед, то це тварина хижак;
- Якщо тварина ссавець і має копита, то це тварина парнокопитна;
- Якщо тварина ссавець і жує жуйку, то це тварина парнокопитна;
- Якщо тварина ссавець і це тварина хижак і це тварина жовто-коричневого кольору і це тварина має темні плями, то це тварина гепард;
- Якщо тварина ссавець і це тварина хижак і це тварина жовто-коричневого кольору і це тварина має темні смуги, то це тварина тигр;
- Якщо тварина є парнокопитною і має довгу шию і має довгі ноги і має чорні плями, то це тварина жираф;
- Якщо тварина є парнокопитною і має чорні смуги, то це тварина зебра.

29. Уявимо собі, що був урожай і господиня наготувала багато варення. Залишилося розлити його по банках - одно-, двох- і трилітровим. Складіть програму на Пролозі, яка за загальним об'ємом варення з'ясовувала б - скільки та яких банок буде потрібно.

30. У глек А вміщається 5 літрів, а в глек В входить 2 літри. При старті 5-літровий глек повний, ви можете лити воду з одного глека в інший або на землю, повністю наповнювати глеки водою. Це робиться до тих пір, поки ви не упевнені, що глек В містить рівно 1 літр. Складіть програму на Пролозі такого розливу.

31. Гравець А вибирає секретний код, що є послідовністю N різних десяткових цифр (зазвичай N встановлюється рівним 4). Гравець В намагається вгадати задуманий код і питає гравця А про число "биків" (число "биків" - кількість співпадаючих цифр в однакових позиціях передбачуваного і задуманого коду; число "корів" - кількість співпадаючих цифр, що входять в передбачуваний і задуманий код, але що знаходяться на різних позиціях).

Код вгаданий якщо число биків рівне N.

Алгоритм загалом описується таким чином.

Вводиться деякий порядок на множині допустимих правилами пропозицій; висунення чергових припущень враховує накопичену на той час інформацію, і так до тих пір, поки секретний код не буде розкритий.

Розглянутий алгоритм дозволяє вирішити задачу - розкрити код за 4-6 спроб (як досвідчений гравець). Складіть програму на Пролозі цього алгоритму.

32. Скласти графічну версію програми "Ханойські вежі" на Visual Prolog.

33. Необхідно визначити, чи є даний об'єкт автомобілем або мотоциклом. У автомобіля є колеса і дверці. У мотоцикла є руль і колеса. У автомобіля і мотоцикла є двигун. Уточнюючи всі ці характеристики, ми повинні визначити об'єкт. Сформулюйте і запишіть ці правила у вигляді продукції.

34. Побудувати мережу фреймів, що складається з понять: море, корабль, матрос.

35. Зобразити семантичну мережу ситуації «Студент на лекції», де як вершини виступають поняття: Викладач, Студент, Аудиторія, Конспект, Сусід, Тема лекції, Питання, Доска, Курс, Дисципліна, Назва факультету, Група. Використовуючи ознаки класифікації семантичних мереж, визначити її видові характеристики.

36. Біля дверей кімнати перебуває мавпа. В середині цієї кімнати до стелі підвішений банан. Мавпа голодна і хоче з'їсти банан, проте вона не може дотягнутися до нього, знаходячись на підлозі. Біля вікна цієї ж кімнати на підлозі лежить ящик, яким мавпа може скористатися. Мавпа може робити наступні дії: ходити по підлозі, залізати на ящик, рухати ящик (якщо вона вже знаходиться біля нього) і схопити банан, якщо вона стоїть на ящику прямо під бананом. Скласти программу на Пролозі, яка б вказувала послідовність дій обез’яни.

37. Скласти программу на Пролозі, яка б знаходила таку розстановку восьми ферзів на порожній шахівниці, в якій жоден з ферзів не знаходиться під боєм іншого.

2. Завдання до індивідуальної роботи

1. Взаємний зв'язок між базами даних та базами знань.
2. Логічні методи представлення знань та їх характеристики.
3. Представлення пошукових просторів у вигляді графів.
4. Двонаправлений пошук розв’язків.
5. Правило факторизації і метод резолюції.
6. Поняття складності логічної програми та основні підходи до її вимірювання.
7. Особливості та характеристики систем штучного інтелекту.
8. Структура програми на Пролозі.
9. Декларативна та процедурна семантика Пролог-програм.
10. Механізми пошуку знань в експертній системі.
11. Проблеми видобування інформації від експертів.
12. База знань інструментальної оболонки ЕС CLIPS.
13. База знань інструментального комплекса ЕС реального часу G2.
14. Представлення знань в інструментальному комплексі ЕС EXSYS.
15. Моделювання недетермінованого автомата на мові Пролог.
16. Представлення графів на мові Пролог.
17. Побудова остовного дерева на мові Пролог.
18. Представлення множин двійковими деревами на мові Пролог.
19. Представлення двійкових довідників на мові Пролог.
20. Базові процедури пошуку в і/або-графах на мові Пролог.
21. Розробка оболонки експертної системи на мові Пролог.
22. Реализація гри двох осіб з повною інформацією на мові Пролог.
23. Прологівські програми як системи, що керуються зразками.
24. Автоматичне доведення теорем на мові Пролог.
25. Стратегія пошуку в глибину і ширину на мові Пролог.
26. Використання фреймів в евристичному пошуку.
27. Засоби представлення знань в Семантичній мережі.
28. Експертні системи в WWW.
29. Класи і об'єкти Visual Prolog.
30. Вбудовані предикати мови Visual Prolog.
31. Схеми обробки списків у Visual Prolog.
32. Типи даних Visual Prolog.
33. Використання нечітких множин при логічному виводі
34. Нечіткі прямий і зворотний виводи.
35. Числення нечітких величин.
36. Теорія Демпстера - Шаффера і чинники упевненості.
37. Представлення системи продукций “І/або” графом. Вивід за наявності нечіткої інформації.
38. Управління виводом в продукційній системі. Установка обмежень на генерацію конфліктного набору. Вивід по пріоритету глибини. Проблеми реалізації стратегій пошуку виводу.

Документація ПМК

з курсу “МЕТОДИ ТА ЗАСОБИ ПОДАННЯ ЗНАНЬ”

ПОЯСНЮВАЛЬНА ЗАПИСКА

Пакет контрольних завдань з курсу «Методи та засоби подання знань», складений згідно програми курсу і освітньо-кваліфікаційних вимог до підготовки бакалаврів за спеціальністю 6.080400 «Інтелектуальні системи прийняття рішень».

Пакет завдань включає в себе 3 частини:

1- модульна контрольна робота №1;

2- модульна контрольна робота №2;

3- комплексна контрольна робота для підсумкового контролю.

Вимоги до виконання завдань та оцінка

Оцінка підсумкового контролю знань студентів – *чотирибальна* /«відмінно», «добре», «задовільно», «незадовільно»/.

Критерії оцінки знань

Критерії оцінки підсумкового контролю знань студентів базуються на навчальній програмі, робочому плані та найбільш важливих умовах до знань студентів:

1. знання фактів, явищ і вірне, науково достовірне їх пояснення;
2. оволодіння науковими термінами, поняттями, законами, методами, правилами; вміння користуватися ними при вирішенні різних питань і виконанні практичних завдань;
3. максимальна ясність, точність думки;
4. знання повинні мати практичну значимість; студенти повинні вміти безпосередньо застосувати їх на комп'ютері.

Відповіді на теоретичні питання повинні бути повними, логічними, доведеними. Практичні завдання студентів повинні бути виконані з точним дотриманням вказівок викладача.

На оцінку «відмінно» відповідь студента повинна відповідати пунктам 1-4, на «добре» – 1, 2, 4, на «задовільно» 1,4.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____1_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Поняття знання та моделі подання знань 2. Метод резолюції 3. Задача. Мовою Пролог-Д напишіть базу знань, у якій визначається функція Хевисайда.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____2_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Дескриптивний та процедурний зміст програми на мові Пролог 2. Основи аксіоматичних систем та числення предикатів 3. Задача. Напишіть програму на Пролозі, що знаходить ім'я матері хлопчика.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____3_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Подання знань у формі клауз 2. Арифметичні й інші убудовані предикати мови Пролог 3. Задача. Написати мовою Пролог базу знань, що описує обчислення факторіала.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____4_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Поняття рекурсії й структури даних у програмах на мові Пролог 2. Подання знань про предметну область у вигляді фактів й правил 3. Задача. Написати мовою Пролог базу знань, що описує обчислення суми чисел натурального ряду від 1 до N.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____5_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Алгоритми обробки списків на мові Пролог 2. Історія розвитку поняття знання у комп'ютерних системах 3. Задача. Написати мовою Пролог базу знань, що описує обчислення суми квадратів чисел натурального ряду від 1 до N.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____6_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Загальна характеристика моделей подання знань у пам'яті інтелектуальної системи (логічних, мережевих, продукційних, фреймових тощо) 2. Аксіоматика та правила виводу числення предикатів 3. Задача. Описати обчислення найменшого загального кратного.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____7_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Моделювання інтелектуальної діяльності людини за допомогою формально-логічних систем (числення висловлювань, числення предикатів тощо) 2. Поняття клауз Хорна. Співвідношення між клаузальною формою та стандартною формою логіки 3. Задача. Напишіть мовою Пролог базу знань, що описує прямокутний трикутник.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____8_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Процедурна інтерпретація клауз Хорна. Основи пошуку розв'язків на мові клауз Хорна 2. Негативні цілі та твердження. Поняття підстановки та уніфікації виразів 3. Задача. Скласти програму на Пролозі обчислення суми елементів заданого числового списку.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** № 9 Методи та засоби подання знань (предмет) Група Семестр_____	
1. Загальне правило резолюцій. Глобальні стратегії пошуку розв'язків 2. Зв'язок між логікою й програмуванням. Основний принцип використання мови Пролог 3. Задача. Скласти програму на Пролозі обчислення середнього арифметичного елементів заданого числового списку.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** № 10 Методи та засоби подання знань (предмет) Група Семестр_____	
1. Принципова відмінність Прологу від традиційних мов програмування 2. Поняття бази знань у системі Пролог 3. Задача. Скласти програму на Пролозі, замінюючи в заданому списку всі негативні числа нулями.		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____11_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Програмування на Пролозі як процес створення системи фактів і правил, що характеризують розв'язуване завдання 2. Убудований арифметичний предикат МНОЖЕННЯ 3. Задача. Скласти програму на Пролозі «Ханойська вежа»		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____12_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Побудова бази знань як процес виявлення множини досліджуваних об'єктів і зв'язків між ними 2. Предикати БІЛЬШЕ й НЕ та їх застосування 3. Задача. Скласти програму на Пролозі «Словник»		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____13_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Створення інформаційно-логічної моделі, що описується мовою Пролог 2. Убудований предикат "відсікання" для керування логічним виводом 3. Задача. Скласти програму на Пролозі «Родина»		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____14_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Методики та приклади розробки баз знань у вигляді множини фактів і правил 2. Вплив порядку речень у базі знань на результат виконання рекурсивної програми 3. Задача. Скласти програму на Пролозі «Найбільший загальний дільник двох чисел»		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____15_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Поняття та приклади списків. Типові операції над списками 2. Синтаксис Прологу. Константи. Змінні. Структури. 3. Задача. Скласти програму на Пролозі «Об'єм конуса»		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____16_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Диз'юнкція цілей Пролог-програм 2. Формальний опис процедури обчислення цілей 3. Задача. Скласти програму на Пролозі з оператором АБО		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____17_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Типові структури даних у програмах на мові Пролог 2. Приклади баз знань на мові Пролог, що обробляють спискові структури 3. Задача. Скласти програму на Пролозі з оператором НЕ		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____18_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Створення інформаційно-логічної моделі, що описується мовою Пролог 2. Особливості моделювання інтелектуальної діяльності людини за допомогою формально-логічних систем 3. Задача. Скласти програму на Пролозі з оператором відсікання		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____19_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Особливості та характеристики систем штучного інтелекту 2. Взаємний зв'язок між базами даних та базами знань 3. Задача. Скласти програму на Пролозі з оператором порівняння		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____20_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Створення база знань для виконання арифметичних операцій ДОДАВАННЯ(X,Y,Z), ВІДНІМАННЯ(X,Y,Z), МНОЖЕННЯ(X,Y,Z), ДІЛЕННЯ(X,Y,Z) 2. Логічні основи мови Пролог (терми, функтори, предикати, логічні зв'язки, логічні формули, резолюції тощо) 3. Задача. Наведіть приклад головоломки на мові Пролог		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____21_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Убудований предикат "відсікання" для керування логічним виводом 2. Синтаксис Прологу. Константи. Змінні. Структури 3. Задача. Скласти програму на Пролозі для обчислення площі кола		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____22_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Що таке продукційна модель подання знання? 2. Поняття підстановки та уніфікації виразів 3. Задача. Скласти програму на Пролозі для обчислення довжини кола		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____23_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Типові структури даних у програмах на мові Пролог 2. Приклади баз знань на мові Пролог, що обробляють спискові структури 3. Задача. Запропонуйте правило на мові Пролог, що пов'язує дохід і податок на нього		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____24_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Диз'юнкція цілей Пролог-програм 2. Формальний опис процедури обчислення цілей 3. Задача. Для фактів виду УЧЕНЬ(прізвище, предмет, оцінка) запропонуйте правила ВСТИГАЮЧИЙ(прізвище), НЕВСТИГАЮЧИЙ(прізвище), ВІДМІННИК(прізвище)		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____25_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Загальне правило резолюцій. Глобальні стратегії пошуку розв'язків 2. Зв'язок між логікою й програмуванням. Основний принцип використання мови Пролог 3. Задача. Для фактів виду АВТОМОБІЛЬ(марка, країна, рік) запропонуйте правила НОВИЙ(марка), СТАРИЙ(марка), ІНОМАРКА(марка), ВІТЧИЗНЯНИЙ(марка).		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** №_____26_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Процедурна інтерпретація клауз Хорна. Основи пошуку розв'язків на мові клауз Хорна 2. Негативні цілі та твердження. Поняття підстановки та уніфікації виразів 3. Задача. Для фактів виду МІСТО(назва, населення) запропонуйте правила МІСТЕЧКО(назва), СЕРЕДНЄ(назва), МЕГАПОЛІС(назва).		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** № 27 Методи та засоби подання знань (предмет) Група Семестр_____	
1. Методики та приклади розробки баз знань у вигляді множини фактів і правил 2. Вплив порядку речень у базі знань на результат виконання рекурсивної програми 3. Задача. Напишіть програму на Пролозі, що знаходить ім'я батька хлопчика		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** № 28 Методи та засоби подання знань (предмет) Група Семестр_____	
1. Алгоритми обробки списків на мові Пролог 2. Історія розвитку поняття знання у комп'ютерних системах 3. Задача. Напишіть програму на Пролозі, що знаходить ім'я діда хлопчика		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** № 29 Методи та засоби подання знань (предмет) Група Семестр_____	
1. Алгоритм уніфікації формул логіки першого порядку 2. Метод сортування списків «бульбашки» 3. Задача. Побудуйте базу знань, використовуючи правила: • Якщо тварина має волосся, то ця тварина - ссавець; • Якщо тварина має пір'я, то ця тварина - птах; • Якщо тварина може літати і відкладає яйця, то ця тварина - птах; • Якщо тварину їсть м'ясо, то ця тварина - хижак		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №1** № 30 Методи та засоби подання знань (предмет) Група Семестр_____	
1. Структура програми на мові Пролог 2. Механізм backtracking у мові Пролог 3. Задача. Напишіть програму на Пролозі, яка для заданого списку визначає скільки разів в ньому зустрічається заданий елемент		

"_____" _____ 2009 р.

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____1_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Однорідні та неоднорідні семантичні мережі 2. Означення лінгвістичної змінної 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____2_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Концепція семантичної павутини 2. Поняття експертної системи (ЕС). Сфера застосування ЕС 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____3_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Формат RDF подання знань 2. Структура експертної системи та її основні компоненти 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____4_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Мова OWL (Web Ontology Language) подання знань 2. Основні режими роботи експертних систем 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____5_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Засоби представлення даних та знань в Семантичній мережі 2. Етапи розробки ЕС 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____6_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Мережні продукції 2. Особливості ЕС, що відрізняють їх від звичайних програм. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____7_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Семантичні мережі, що задають значення 2. Визначення складу та моделі представлення знань в ЕС. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____8_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Семантичні графи простого виду 2. Рівні представлення і рівні детальності знань в ЕС. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____9_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Структура фрейму 2. Організація знань в робочої пам'яті експертної системи. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____10_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Базові елементи фреймів 2. Подання знань в експертній оболонці EsWin. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №____11_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Визначення фрейму в нотації Бекуса-Наура 2. Поняття метазнання ЕС. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №____12_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Фрейми-структури 2. Основні етапи реалізації системи придбання знань. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №____13_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Фрейми-ролі 2. Метод протокольного аналізу виявлення процедурних знань. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №____14_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Фрейми-сценарії 2. Методи пошуку в одному просторі. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____15_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Фрейми-ситуації 2. Методи пошуку в ієрархічних просторах.. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____16_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Мережа фреймів 2. Методи пошуку при неточних і неповних даних. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №____17____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Глобальна система просторових фреймів 2. Засоби представлення знань і стратегії управління статичних експертних систем. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №____18____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Продукційні моделі бази знань 2. Характеристика інструментальної оболонки CLIPS. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____19_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Продукції типу AW=>BR 2. Можливості оболонки експертної системи GURU. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____20_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Продукції типу AW=>BK 2. Інструментальні комплекси для створення ЕС реального часу. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____21_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Продукції типу АК=>BW 2. Сучасні тенденції розвитку штучного інтелекту. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____22_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Що таке продукційна модель подання знання? 2. Представлення знань в оболонці експертної системи G2. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____23_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Пошук розв'язків продукційної моделі 2. Застосування чинників упевненості та нечітких змінних при побудові ЕС з неточними і неповними даними 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____24_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Поняття недостовірної та нечіткої інформації 2. Особливості архітектури експертної системи реального часу. 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____25_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Функція належності та операції над нечіткими множинами 2. Формування набору правил та бази знань ЕС 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол № від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____26_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Нечітке логічне виведення 2. Основні компоненти експертної системи реального часу 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №____27_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Сфери застосування нечіткої логіки 2. Загальна характеристика командної мови "GURU" 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №____28_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Перетин двох нечітких множин 2. Електронні таблиці оболонки ЕС "GURU". 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____29_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Об'єднання двох нечітких множин 2. Механізми пошуку знань в експертній системі 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри ІСПР Протокол №__ від_____ Зав. кафедрою _____	**МОДУЛЬНА КОНТРОЛЬНА РОБОТА №2** №_____30_____ Методи та засоби подання знань (предмет) Група Семестр_____	
1. Означення нечіткої змінної 2. Інструментальні комплекси для побудови статичних ЕС 3. Задача.		

"_____" _____ 2009 р.

Науково-педагогічний працівник_____

Перелік завдань
для модульної контрольної робота №2
з дисципліни "Методи та засоби подання знань"

1. Скласти фрейм з ім'ям Книга, де іменами слотів є Автор, Назва книги, Об’єм, Видавництво, Рік видання.

2. Скласти фрейм з ім'ям Курс, де іменами слотів є Номер курсу, Кількість студентських груп, Куратор, Кафедра, Старости груп.

3. Скласти фрейм з ім'ям Дисципліна, де іменами слотів є Назва дисципліни, Кількість годин, Література, Викладач, Підсумковий контроль.

4. Скласти фрейм з ім'ям Комп’ютер, де іменами слотів є Операційна система, Процесор, Оперативна пам’ять, Материнська плата, Зовнішня пам’ять.

5. Скласти фрейм з ім'ям Потяг, де іменами слотів є Номер потяга, Вагон, Купе, Вартість квітка, Час відправлення.

6. Сформулюйте і запишіть правила, які визначають, чи буде сьогодні дощ, у вигляді продукції. Спочатку ви визначаєте, чи ясне небо. Якщо небо ясне, то дощу не буде. Якщо небо похмуре, то ви дивитеся, чи є на небі чорні грозові хмари. Якщо немає, то дощу не буде. Якщо є, то дивитеся, в яку сторону вони рухаються. Якщо у вашу сторону, то дощ буде. Якщо ж ні - то не буде.

7. Нехай експерт визначає вагу виробу за допомогою понять "Мала вага", "Середня вага" і "Велика вага", при цьому мінімальна вага дорівнює 1 кг, а максимальна – 10 кг. Формалізувати поняття ваги виробу.

8. Нехай нечіткі множини A, B, C задані функціями належності:

$$A = 0{,}6/x_1 + 0{,}1/x_2 + 0{,}5/x_3 + 1/x_4$$

Виконати логічні операції над цими множинами.

9. Нехай нечіткі множини A, B, C задані функціями належності:

$$B = 0{,}2/x_1 + 0{,}5/x_2 + 0{,}7/x_3 + 1/x_4$$

Виконати логічні операції над цими множинами.

10. Нехай нечіткі множини A, B, C задані функціями належності:

$$C = 0{,}5/x_1 + 0{,}3/x_2 + 0{,}2/x_3 + 0{,}6/x_4$$

Виконати логічні операції над цими множинами.

11. Нехай експерт визначає термін придатності виробу за допомогою понять "Малий термін придатності", "Середній термін придатності" і "Великий термін придатності", при цьому мінімальний термін придатності дорівнює 1 рік, а максимальний – 15 років. Формалізувати поняття термін придатності виробу.

12. Нехай експерт визначає трудомісткість виробу за допомогою понять "Мала трудомісткість", "Середня трудомісткість" і "Велика

трудомісткість", при цьому мінімальна трудомісткість дорівнює 120 роб. год., а максимальна – 560 роб. год. Формалізувати поняття трудомісткість виробу.

13. Нехай експерт визначає висоту виробу за допомогою понять "Мала висота", "Середня висота" і "Велика висота", при цьому мінімальна висота дорівнює 1 м, а максимальна – 5 м. Формалізувати поняття висоти виробу.

14. Нехай експерт визначає довжину виробу за допомогою понять "Мала довжина", "Середня довжина" і "Велика довжина", при цьому мінімальна довжина дорівнює 20 см, а максимальна – 65 см. Формалізувати поняття довжини виробу.

15. Нехай експерт визначає вартість виробу за допомогою понять "Мала вартість", "Середня вартість" і "Велика вартість", при цьому мінімальна вартість дорівнює 150 грн., а максимальна – 2000 грн. Формалізувати поняття вартість виробу.

16. Нехай нечіткі множини A, B, C задані функціями належності:

$$A = 0{,}4/x_1 + 0{,}2/x_2 + 0/x_3 + 1/x_4$$

Виконати логічні операції над цими множинами.

17. Нехай нечіткі множини A, B, C задані функціями належності:

$$C = 0{,}1/x_1 + 1/x_2 + 0{,}2/x_3 + 0{,}9/x_4$$

Виконати логічні операції над цими множинами.

18. Нехай нечіткі множини A, B, C задані функціями належності:

$$A = 0{,}1/x_1 + 0{,}5/x_2 + 0{,}8/x_3 + 0{,}1/x_4$$

Виконати логічні операції над цими множинами.

19. Нехай нечіткі множини A, B, C задані функціями належності:

$$B = 0{,}6/x_1 + 0{,}4/x_2 + 0{,}5/x_3 + 0{,}6/x_4$$

Виконати логічні операції над цими множинами.

20. Сформулюйте і запишіть правила, що визначають приймати чи ні людини на роботу, у вигляді продукції. Якщо людина не має вищої освіти, то відмовити. Якщо має, то скільки років пропрацював претендент за фахом. Якщо менше року, то відмовити, якщо більше - то прийняти. Якщо претендент має вчене звання, то запропонувати посаду наукового співробітника, якщо немає, то посада інженера-конструктора.

21. Сформулюйте і запишіть правила, які дають поради при створенні і відладці програми, у вигляді продукції. Якщо ви написали програму на мові високого рівня і при компіляції виявили синтаксичні помилки, то необхідно виправити програму. Якщо помилок немає, то необхідно запустити редактора зв'язків. Якщо редактор зв'язків повідомляє про помилки, то необхідно перевірити наявність всіх початкових модулів. Якщо помилок при лінкувані немає, то програма готова до роботи.

22. Сформулюйте і запишіть правила, що визначають, чи є у дитини труднощі при вивченні арифметики, у вигляді продукції. Якщо у дитини проблеми при вивченні складання, то вона зазнає труднощі при вивченні арифметики. Якщо дитина має проблеми при вивченні множення, то вона зазнає труднощі при вивченні арифметики. Аналогічно з відніманням і діленням.

23. Ви бажаєте прогнозувати на біржі рівень цін. Якщо валютний курс долара падає, то процентні ставки зростають. Якщо валютний курс долара зростає, то процентні ставки падають. Якщо процентні ставки зростуть, то рівень цін на біржі упадає. Якщо процентні стави упадають, то рівень цін на біржі зросте. Сформулюйте і запишіть ці правила у вигляді продукції.

24. Зобразити семантичну мережу, де як вершини виступають поняття: Людина, Шевченко, Таврія, Автомобіль, Вид транспорту, Двигун. Використовуючи ознаки класифікації семантичних мереж, визначити її видові характеристики.

25. Зобразити семантичну мережу ситуації «Отримання студентом книги в бібліотеці», де як вершини виступають поняття: Книга, Студент, Бібліотека, Назва бібліотеки, Автор книги, Назва книги, Місце розташування бібліотеки. Використовуючи ознаки класифікації семантичних мереж, визначити її видові характеристики.

26. Побудувати мережі фреймів, що складається з понять: Людина, Дитина, Учень.

27. Побудувати мережі фреймів, що складається з понять: Транспорт, Автомобіль, Запорожець.

28. Побудувати мережі фреймів, що складається з понять: Університет, факультет, група.

29. Побудувати мережі фреймів, що складається з понять: Установа, відділ, відділення.

30. Побудувати мережі фреймів, що складається з понять: Математика, алгебра, теорія матриць.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_1_ з курсу «Методи та засоби подання знань»
1. Загальна характеристика моделей подання знань у пам'яті інтелектуальної системи. 2. Визначення та класифікація семантичних мереж (процедурні, розподілені, функціональні, фреймові, нейроні, концептуальні, ситуаційні). 3. Опишіть мовою логіки першого порядку властивості операції додавання, множення.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_2_ з курсу «Методи та засоби подання знань»
1. Формальні теорії та аксіоматичні системи. Моделювання інтелектуальної діяльності людини за допомогою формально-логічних систем. 2. Способи завдання семантичних мереж. Види семантичних відносин: ієрархічні, функціональні, кількості, просторові, часові, атрибутні, логічні, лінгвістичні. 3. Опишіть мовою логіки першого порядку властивості відносини рівність.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_3_ з курсу «Методи та засоби подання знань»
1. Подання знань у формі клауз. 2. Інтернет як розподілена база знань та семантична мережа глобального масштабу. Використання семантичних мереж у системах штучного інтелекту (системах машинного перекладу, експертних системах тощо). 3 Опишіть мовою Пролог склад своєї родини.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_4_ з курсу «Методи та засоби подання знань»
1. Знання як об’єкти комп’ютерної обробки. Метод резолюцій. 2. Особливість фрейм-підходу до проблеми подання знань. Поняття, структура та властивості фреймів (слоти, процедури та правила, наслідування, статичність та динамічність). 3. Складіть базу знань, що описує країни Європи, Азії, інших материків.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_5_ з курсу «Методи та засоби подання знань»
1. Зв'язок між логікою й програмуванням. Основний принцип використання мови Пролог. 2. Класифікація фреймів (фрейм-образ, фрейм-сценарій). Конкретизація, ієрархія та наслідування фреймів. Фрейми та об'єктно-орієнтоване програмування. 3. Напишіть мовою Пролог таблицю множення чисел від 1 до 10. Яка кількість речень потрібно для запису цієї бази знань?	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_6_ з курсу «Методи та засоби подання знань»
1. Поняття бази знань у системі Пролог. 2. Поняття та властивості продукційної моделі. Загальна структура продукції та її компоненти. 3. Опишіть мовою Пролог обчислення площ геометричних фігур: трапеції, трикутника, паралелограма.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_7_ з курсу «Методи та засоби подання знань»
1. Подання знань про предметну область у вигляді фактів й правил. 2. Процедури управління системою продукції. Класифікація ядер продукції. Стратегії організації пошуку розв'язків. 3. Опишіть обчислення площі кругу й довжини окружності. Яка точність обчислень цих величин? Чи можна обчислити радіус кола по довжині окружності?	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_8_ з курсу «Методи та засоби подання знань»
1. Арифметика й інші убудовані предикати мови Пролог. 2. Типова схема роботи експертної системи на базі продукції. Пряме та зворотне виведення. 3. Мовою Пролог напишіть базу знань, у якій визначається функція Хевисайда.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_9_ з курсу «Методи та засоби подання знань»
1. Рекурсія та структури даних у програмах на мові Пролог. 2. Поняття недостовірної та нечіткої інформація. Об'єктивна та суб'єктивна невизначеність. Модальна логіка. 3. Які складності можуть виникнути в базі знань про матерів, якщо їх діти будуть тезками?	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_10_ з курсу «Методи та засоби подання знань»
1. Обробка списків на мові Пролог. 2. Функція належності та основні операції над нечіткими множинами. 3. Напишіть програму на Пролозі, що знаходить ім'я матері хлопчика.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_11_ з курсу «Методи та засоби подання знань»
1. Синтаксис Прологу. Константи. Змінні. Структури. Оператори. 2. Функція належності та основні операції над нечіткими множинами. 3. Написати мовою Пролог базу знань, що описує обчислення факторіала.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_12_ з курсу «Методи та засоби подання знань»
1. Декларативна семантика Пролог-програм. 2. Неточне логічне виведення. Принципи неточного виведення. 3. Написати мовою Пролог базу знань, що описує обчислення суми чисел натурального ряду.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_13_ з курсу «Методи та засоби подання знань»
1. Процедурна семантика Пролог-програм. 2. Сфера застосування ЕС. Структура експертної системи та її основні компоненти. 3. Написати мовою Пролог базу знань, що описує обчислення суми квадратів чисел натурального ряду.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_14_ з курсу «Методи та засоби подання знань»
1. Логічні методи представлення знань та їх характеристики. 2. Подання знань в ЕС. Визначення складу та моделі представлення знань в ЕС. 3. Описати обчислення найменшого загального кратного.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_15_ з курсу «Методи та засоби подання знань»
1. Програмування на Пролозі як процес створення системи фактів і правил, що характеризують розв'язуване завдання. 2. Характеристика бази знань ЕС. Механізми пошуку знань в експертній системі. 3. Напишіть мовою Пролог базу знань, що описує прямокутний трикутник.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_16_ з курсу «Методи та засоби подання знань»
1. Створення інформаційно-логічної моделі, що описується мовою Прологу. 2. Інструментальна оболонка "GURU": режими роботи, правила, команди, стратегія управління. 3. Використовуючи рекурсивне визначення, напишіть базу знань, що описує багатоповерховий будинок.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_17_ з курсу «Методи та засоби подання знань»
1. Убудований предикат "відсікання" для керування логічним виводом. 2. Експертні системи реального часу як основний напрям розвитку систем штучного інтелекту. 3. Опишіть мовою Пролог побудову вулиці без урахування та з урахуванням перспективи.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_18_ з курсу «Методи та засоби подання знань»
1. Поняття та приклади списків. Типові операції над списками: приналежність елемента до списку, склеювання двох списків, сортування та обертання списку, знаходження та видалення елемента списку. 2. Характеристика оболонок експертних систем реального часу як самодостатніх середовищ для розробки, впровадження і супроводу застосувань в широкому діапазоні галузей. 3. Скласти програму на Пролозі обчислення суми елементів заданого числового списку.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_19_ з курсу «Методи та засоби подання знань»
1. Формальний опис процедури обчислення цілей у мові «Пролог» 2. Універсальні технології побудови сучасних ЕС (стандарти відкритих систем, архітектура клієнт/сервер, об'єктно-орієнтоване програмування). 3. Скласти програму на Пролозі обчислення середнього арифметичного елементів заданого числового списку.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_20_ з курсу «Методи та засоби подання знань»
1. Основи аксіоматичних систем. Числення предикатів. 2. Спеціалізовані методи ЕС (міркування, засновані на правилах, міркування, засновані на динамічних моделях, імітаційне моделювання, процедурні міркування, структурована природна мова для представлення бази знань). 3. Скласти програму на Пролозі, що породжує список, що містить натуральні числа від 1 до N (за збільшенням).	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_21_ з курсу «Методи та засоби подання знань»
1. Історія розвитку поняття знання у комп'ютерних системах (поняття бази знань як розвиток поняття бази даних). 2. Мова команд оболонки "GURU". Команди BOILD, COMPILE, CONSULT, RUN, DIR, LET, OUTPUT, INPUT, PERFORM, RETURN, WAIT, WHILE-DO, TEST, IF-THEH-ELSE, CONTINUE, BREAK, CLEAR, BYE. 3. Скласти програму на Пролозі, що породжує список, що містить квадрати натуральних чисел від 1 до N (за збільшенням).	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_22_ з курсу «Методи та засоби подання знань»
1. Недоліки представлення знання за допомогою аксіоматичних систем та область застосування формально-логічних методів. 2. Режими обробки електронних відомостей "GURU". Команди EB\DUMP, EB\DISPLAY, EB\STOP, EB\LOAD, EB\SAVE, EB\COPY, EB\LET, EB\COMPUTE, EB\WIDTH, EB\USING, EB\UNDEFINE, EB\EDIT. 3. Скласти програму на Пролозі, що дозволяє видалити із заданого списку всі від'ємні числа.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_23_ з курсу «Методи та засоби подання знань»
1. Негативні цілі та твердження. Поняття підстановки та уніфікації виразів. Глобальні стратегії пошуку розв'язків. 2. Графічні засоби оболонки "GURU" 3. Скласти програму на Пролозі, замінюючи в заданому списку всі негативні числа нулями.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_24_ з курсу «Методи та засоби подання знань»
1. Особливості та характеристики систем штучного інтелекту. 2.Основні кроки алгоритму створення ЕС. 3. Скласти програму на Пролозі, замінюючи в заданому списку всі числа їх квадратами.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_25_ з курсу «Методи та засоби подання знань»
1. Поняття клауз Хорна. Співвідношення між клаузальною формою та стандартною формою логіки. Основи пошуку розв'язків на мові клауз Хорна. 2. Класифікація методів видобування знань. Проблеми видобування інформації від експертів. 3. Скласти програму на Пролозі, яка повідомляє, що даний елемент є N-м в заданому списку.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_26_ з курсу «Методи та засоби подання знань»
1. Програмування математичних задач в системі Пролог. Створення база знань для виконання арифметичних операцій ДОДАВАННЯ(X,Y,Z), ВІДНІМАННЯ(X,Y,Z), МНОЖЕННЯ(X,Y,Z), ДІЛЕННЯ(X,Y,Z). 2. Вирази та функції оболонки "GURU". 3. Скласти програму на Пролозі, що замінює в заданому списку два однакові елементи, що стоять поряд, одним.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_27_ з курсу «Методи та засоби подання знань»
1. Програмування основних операції над списками на прикладі баз знань, що описують: сортування списку, обертання списку, знаходження елемента списку, видалення елемента списку. 2. Засоби для визначення релевантних знань. Зв'язність знання та даних, механізм доступу до знань та синтаксична, параметрична, семантична відповідність. 3. Скласти програму на Пролозі, що встановлює, що два задані списки не мають загальних елементів.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_28_ з курсу «Методи та засоби подання знань»
1. Основні об'єкти мови Пролог: формули, речення, диз'юнкти, диз'юнкти Хорна, факти, правила, питання, порожній диз'юнкт. 2. Статичні та динамічні ЕС. Етапи розробки ЕС (ідентифікація, концептуалізація, формалізація, виконання, тестування, дослідна експлуатація. 3. Скласти програму на Пролозі, що встановлює, що один список є підсписком іншого, тобто всі елементи одного списку, містяться в іншому списку.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_29_ з курсу «Методи та засоби подання знань»
1. Створення інформаційно-логічної моделі, що описується мовою Прологу. Методики та приклади розробки баз знань у вигляді множини фактів і правил. 2. Визначення складу та моделі представлення знань в ЕС. Знання, необхідні для ЕС (знання про процес розв'язування задачі, про мову спілкування та способах організації діалогу, про способи представлення та модифікації знань, підтримуючи структурні та управлінські знання, знання про методи взаємодії із зовнішнім середовищем, знання про модель зовнішнього світу). Інтерпретація знання. 3. Скласти програму на Пролозі, що дозволяє розділити заданий список на два підсписки, в перший помістити всі позитивні числа з початкового списку, в другій - від'ємні, а нулі - відкинути.	

Науково-педагогічний працівник _____

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ
(м. Ірпінь Київської області)

Розглянуто і затверджено на засіданні кафедри інтелектуальних систем прийняття рішень Протокол №___ від «___» _____200__р. Зав. кафедрою_____	ККР БІЛЕТ №_30_ з курсу «Методи та засоби подання знань»
1. Представлення знання засобами логіки. Розв'язування логічних задач з використанням числення висловлювань та числення предикатів. 2. Засоби представлення даних та знань в Семантичній мережі. Онтології як засіб представлення знань. 3. Скласти програму на Пролозі, яка по двох заданих списках створює третій, куди поміщає елементи, які присутні в обох списках.	

Науково-педагогічний працівник _____

Перелік рекомендованої літератури

Основна література.

1. Люгер Дж. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем. - М.: Издат. дом “Вильямс”, 2003. - 865 с.
2. Адаменко А.Н., Кучуков А.М. Логическое программирование и Visual Prolog. – СПб.: БХВ – Петербург, 2003. – 992 с.: ил.
3. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПб.: Питер, 2001. – 384 с.: ил.
4. Девятков В.В. Системы искусственного интеллекта. – М.: МГТУ, 2001.
5. Морозов М.Н. Логическое программирование. Курс лекций. 2001. http://www.marstu.mari.ru:8101/mmlab/home/prolog/LECTION1/index.html
6. Осипов Г.С. Приобретение знаний интеллектуальными системами. М.: Наука, 1997. - 360 с.
7. Ин Ц., Соломон Д. Использование Турбо-Пролога: Пер. с англ. - М.: Мир, 1993. - 608 с.
8. Ревунков Г.И. и др. Базы и банки данных и знаний. - М.: Высшая школа, 1992.
9. Лорьер Ж.-Л. Системы искусственного интеллекта // М.: "Мир". 1991. - 342 стр. с илл.
10. Братко И. Программирование на языке Пролог для искусственного интеллекта / Пер. с англ. - М.: Мир, 1990. – 560 с.
11. Ковальски Р. Логика в решении проблем: Пер. с англ. – М.: Наука. Гл. ред. физ.-мат. лит., 1990. – 280 с.
12. ЭС для персональных компьютеров. Методы, средства, реализации. - Минск: Вышейшая школа, 1990.
13. Таусенд К., Фохт Д. Проектирование и программная реализация экспертных систем на персональных ЭВМ / Пер с англ. В.А. Кондратенко. - М.: Финансы и статистика, 1990.
14. Трохимчук Р.М. Збірник задач з дискретної математики. – К.: "Київський університет", 1997.
15. Уотермен Д. Руководство по экспертным системам: пер. с англ./ Под ред. В.Л. Стефанюка. - М.: Мир, 1989
16. Кофман А. Введение в теорию нечетких множеств. - М.:Радио и связь. 1982.- 432 с.
17. Минский М. Фреймы для представления знаний. - М.: Энергия, 1979.
18. Алексеев М.Н., Бешенков С.А., Гейн А.Г., Григорьев С.Г. Информатика и информационные технологии: практические работы. - Миасс, 2000. - 42 с.
19. Eduardo Costa. Visual Prolog 7.1 for Tyros: перевод с английского Сафронов Марк (a.k.a.). 07.08.2007. – 171 с.
20. Лаврентьев В.С., Сергиенко Д.О., Овсянников В.И. Методические указания к лабораторному практикуму "Структуры данных и обработка информационных массивов в экспертных системах". - М.: МИФИ, 2006. - 88 с.

Додаткова література.

1. Достоверный и правдоподобный вывод в интеллектуальных системах / В.Н.Вагин, Е.Ю.Головина, А.А.Загорянский, М.В.Фомина. - М.: Наука. Гл. ред. физ.-мат. лит, 2004. - 704 с.
2. Дюк В.А., Самойленко А. Data mining: Учеб. курс. - СПб.: Питер, 2001. - 366 с.
3. Каллан Р. Основные концепции нейронных сетей. - М.: Издат. дом “Вильямс”, 2001. - 290 с.
4. Карпов О.Н. Технология построения устройств распознавания речи. - Дн.: Изд-во ДНУ, 2001. - 184 с.
5. Алексеев М.Н., Григорьев С.Г. Программирование в системе Пролог-Д (MS-DOS, Windows95/NT). - Миасс, 2000. - 54 с.
6. Андрейчиков А.В., Андрейчикова О.Н., Сергеев С.И. Интеллектуальные информационные системы в экономике. Волгоград. ВГТУ, 1998. - 144 с.
7. Нейроинформатика / А.Н. Горбань и др. - Новосибирск: Наука, 1998.
8. Киселев М., Саломатин Е. Средства добычи знаний в бизнесе и фінансах / Открытые системы, № 4, 1997. стр. 41- 44.
9. Earl Cox. The fuzzy system Handbook. AP.Professional, 1995.
10. Марселлус Д. Программирование экспертных систем на Турбо-Прологе. - М.: Финансы и статистика, 1994.
11. B. Mooge et al. Questions and Answers about G2. 1993. Gensym Corporation. pp. 26-28.
12. B. Moore. Memorandum. 1993, April. Gensym Corparation.
13. Янсон А. Турбо-Пролог в сжатом изложении. - М.: Мир, 1991. - 94 с.
14. Искусственный интеллект: справочник в 3-х книгах. // М.: "Мир", 1990.
15. Логический подход к искусственному интеллекту / Тейз А., Грибомон П., Луи Ж. и др. М.: Мир, 1990. - 432 с.
16. Мелихов А.Н., Берштейн Л.С., Коровин С.Я. Ситуационные советующие системы с нечеткой логикой. - М.: Наука. Гл. ред. физ.-мат. лит., 1990. - 272 с.
17. Поспелов Д.А. Моделирование рассуждений. - М.: Радио и связь, 1989.
18. Выявление экспертных знаний / О.И. Ларичев, А.И. Мечитов и др. - М.: Наука, 1989.
19. Мински М. Фреймы для представления знаний// http://www.compdialog.narod.ru/minski_frame.zip
20. Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web http://www.sciam.com/article.cfm?articleid=000A0919
21. Sheila A. McIlraith, Tran Cao Son, and Honglei Zen. Semantic Web Services, Stanford University. http://www.ksl.stanford.edu.
22. Web Services. http://www-306.ibm.com/software/solutions/webservices/uddi
23. Andon Ph., Deretsky V.. Control Oriented Ontology and Process Description for Cooperation Agents in Information Retrieval// Sixth International Scientific Conference „Electronic Computers and Informatics ECI’2004”. − Kosice – Herlany, Slovakia September 22-24, 2004. P. 14 – 18.