ІНЖЕНЕРІЯ ЗНАНЬ

ЛАБОРАТОРНІ РОБОТИ №2-4

Створення баз знань за допомогою мови подання знань у вигляді фреймів.

1. Мета і завдання лабораторної роботи.

1.1 Метою лабораторної роботи є ознайомлення з фреймовою моделлю подання знань та практичне засвоєння роботи з системою представлення знань ФРЛ.

1.2 Завдання лабораторної роботи:

1. Створення фреймів і робота з ними в системі ФРЛ
2. Створення демонів і приєднаний процедур.
3. Організація мережі фреймів на ФРЛ.
4. Організація бази знань на фреймах в системі ФРЛ.

1.3 Встановлення ФРЛ

Всі питання щодо встановлення останньої версії ФРЛ/ЛІСП можна знайти на github - https://github.com/lispm/FRL .

2. Завдання з мови програмування ФРЛ

2.1 Створення фреймів і вилучення інформації з них

Фрейм представляє собою багаторівневий асоціативний список, що дозволяє описувати різні об'єкти і зв'язки між ними. При цьому кожен слот відображає значення деякої властивості заданого об'єкта. Це значення може бути або явним (аспект **$VALUE**), або заданим за замовчуванням (аспект **$ DEFAULT**). Крім того, значення може бути представлено або в декларативної (описової) формі, або в процедуральній (коментар **STATUS:** зі значенням **EVAL**) формі, застосування якої дозволяє обчислювати значення властивості кожного разу при необхідності його використання. Нижче наводиться приклад створення фрейма і поповнення його додатковою інформацією.

```
( DEFRAMEQ FRAME_1_2_1
           (SLOT_1 ($VALUE (EX1))
                   ($DEFAULT (EX2)) )
           (SLOT_2 ($VALUE (PROC1 (STATUS: EVAL)
                                  (PARM: :FRAME :SLOT))))
       )

( FPRINT  '(FRAME_1_2_1))
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(  FRAME_1_2_1
     ( SLOT_1 2  3($VALUE (EX1))
              ($DEFAULT (EX2))
           )
```

```
        ( SLOT_2 ($VALUE (PROC1 2 ( 3STATUS: EVAL)
                                (PARM: 2  3:FRAME 2  3:SLOT)))
              )
     )

( FASSERTQ FRAME_1_2_1
              (SLOT_3 ($DEFAULT (EX3)))
              (SLOT_1 ($ALUE (EX1) 3  2(EX4)))
              (SLOT_2 ($VALUE (EX5)))
         )

  ( FPRINT  '(FRAME_1_2_1))
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  (  FRAME_1_2_1
     (SLOT_3 ($DEFAULT (EX3))
           )
     ( SLOT_1 2  3($VALUE (EX4) (EX1))
             ($DEFAULT (EX2))
           )
     ( SLOT_2 ($VALUE (EX5)
                     (PROC1 2 ( 3STATUS: EVAL)
                            (PARM: 2  3:FRAME 2  3:SLOT)))
           )
     )
```

1. Створити фрейми, що описують фрагмент бібліотечної системи (що містять як декларативну, так і процедуральних (в тому числі використовує змінні ФРЛ-середовища) складові).
2. Додати у фрейми, певні в попередньому завданні, додаткову інформацію всіма наявними способами.
3. Витягти з визначених у попередніх завданнях фреймів інформацію по заданій множині запитів.
4. Реалізувати функцію послідовного перегляду на екрані фреймів із заданого списку. Передбачити запит про направлення подальшого перегляду списку фреймів.
5. Є фрейми, що описують фрагмент системи з обміну квартир. Реалізувати функцію пошуку інформації про квартирах, які відповідають заданому критерію.

2.2 Приєднані процедури

При роботі з фреймами крім виконання явно вказаних користувачем операцій можуть автоматично виконуватися також і інші, приховані від користувача, операції обробки при виникненні певних ситуацій. Такі ситуації визначаються заздалегідь, і при їх виникненні запускаються приєднані процедури, пов'язані з конкретною ситуацією. Ці своєрідні "демони" можуть виконувати будь-яку додаткову обробку даних, в тому числі і тих, які не входять у фрейм, що містить опис приєднаної процедури. Нижче наводиться приклад завдання приєднаних процедур, автоматично запускаються при видаленні, додаванні або отриманні даних з фрейма. Необхідно зауважити, що дія даних приєднаних процедур поширюється тільки на ті слоти, в яких вони визначені, і не поширюється на інші слоти фрейму. Крім того, в ФРЛ існують два типи функцій, по різному відносяться до приєднаним процедурам. Одні з них активізують приєднані процедури, а інші цього не роблять. Ці ситуації особливо обумовлюються при визначенні вбудованих функцій ФРЛ.

```
    ( DEFRAMEQ FRAME_1_2_2
                  (SLOT_1 ($VALUE (EX1))
                          ($IF-ADDED ((SIGNAL 'ДОДАЄТЬСЯ)))
                          ($IF-REMOVED ((SIGNAL 'УДАЛЯЄТЬСЯ))) )
                  (SLOT_2 ($VALUE (EX2))
                          ($IF-GETED ((SIGNAL 'ВИТЯГУЄТЬСЯ))) )
        )

      ( PASSERTQ SIGNAL (X)
            (PRINT (LIST X " ДАННЕ " :VALUE " З ФРЕЙМА " :FRAME
                                                  " СЛОТА " :SLOT ))
)

      ( FPUT  FRAME_1_2_2  SLOT_1 3  2 $VALUE  'EX3)
            ( ДОДАЄТЬСЯ ДАННЕ  EX3  З ФРЕЙМА  FRAME_1_2_2
                                    СЛОТА  SLOT_1)
          EX3

      ( FPUT  FRAME_1_2_2  SLOT_2 3  2 $VALUE  'EX4)
          EX4

      ( FDELETE  FRAME_1_2_2  SLOT_1  $VALUE  'EX3)
          EX3
 ( FREMOVE  FRAME_1_2_2  SLOT_1  $VALUE  'EX1)
            ( ВИДАЛЯЄТЬСЯ ДАННЕ  EX1  З ФРЕЙМА  FRAME_1_2_2
                                   СЛОТА  SLOT_1)
          EX1
```

6. Забезпечити автоматичний підрахунок частоти звернення до заданого кадру з сигналізацією про кожного кратному 10 зверненні.

7. Місткість вагона 40 т. Товарний склад містить 6 вагонов. Кожен вагон і склад описується своїм фреймом. Крім того, є ще один фрейм, що містить інформацію про кількість та найменування сформованих і відправлених складів. Реалізувати функцію, що моделює завантаження вагона певною кількістю вантажу. При перевищенні місткості вагона вантаж розміщується в наступному вагоні, при формуванні складу інформація про нього включається в головний фрейм і відбувається перехід до формування наступного складу. Реалізувати зазначений алгоритм за допомогою ***$IF-ADDED*** *на всіх рівнях.*

8. Ситуація та ж, що і в попередньому завданні, тільки на станції прибуття потрібно відвантажити задану кількість вантажу. Коригування інформації про наявність вантажу, вагонів і складів реалізувати за допомогою ***$IF-REMOVED****.*

9. Забезпечити автоматичний контроль коректності інформації про наявність на станції прибуття (див. Попереднє завдання) кількості складів, що не перевищують кількість наявних шляхів.

10. У фреймі є слот з закодованим значенням інформації. Забезпечити її витяг по паролю (з використанням ***$IF-NEEDED****).*

2.3. Організація мереж фреймів

Кожен фрейм може являти собою повністю певну закриту одиницю інформації, що задає один конкретний об'єкт. Однак такий підхід при проектуванні і розробці баз знань не може бути визнаний задовільним через те, що це може призвести до дублювання великої кількості інформації, складнощів з підтримкою її поновлення і як наслідок, до отримання

суперечливої інформації. Ця проблема вирішується ефективно в тому випадку, якщо конкретна інфомація (наприклад, про поточну кількість студентів в НТУУ КПІ) міститься лише в одному місці, а всі інші об'єкти, яким ця інформація необхідна, просто посилаються на неї. Такий принцип побудови бази знань називається спадкуванням властивостей, і в ФРЛ він реалізується за допомогою механізму АКО-ієрархії, при якому фрейми пов'язані один з одним в мережу, і пошук необхідної інформації, відсутньої в даному фреймі, здійснюється системою автоматично в усіх інших фреймах, доступних по АКО-ієрархії. Нижче наведено приклад фрагмента мережі фреймів АКО-ієрархії і показані приклади пошуку інформації в ній.

```
    ( DEFRAMEQ FRAME_1_2_3_1
                  (SLOT_1 ($VALUE (EX1)))
                  (SLOT_2 ($VALUE (EX2)))
                  (INSTANCE ($VALUE (FRAME_1_2_3_2)
(FRAME_1_2_3_3)))
           )
      ( DEFRAMEQ FRAME_1_2_3_2
                  (SLOT_3 ($VALUE (EX3)))
                  (SLOT_2 ($VALUE (EX4)))
                  (AKO    ($VALUE (FRAME_1_2_3_1)))
          )

      ( FGET FRAME_1_2_3_2 SLOT_2)
             (EX4 EX2)

      ( FGET1 FRAME_1_2_3_2 SLOT_2)
             (EX4)

      ( FGET1 FRAME_1_2_3_2 SLOT_1)
             (EX1)

      ( FGET FRAME_1_2_3_2 SLOT_4)
             NIL

      ( FGET FRAME_1_2_3_2 SLOT_2 '(C))
             ((EX4 (IN: FRAME_1_2_3_2)) (EX2 (IN: FRAME_1_2_3_1)))
```

11. За допомогою непрямого успадкування задати частина інформації у фрагменті бібліотечної системи (див. Завдання з 2.2).

12. Поставити інформацію про фрагмент бібліотечної системи за допомогою АКО-ієрархії.

13. Є система фреймів, організована в циклічну (кругову) структуру. Забезпечити прохід по циклу задану кількість разів в будь-яку сторону, починаючи з будь-якого фрейму.

14. Сформувати мережеву структуру фреймів з необхідними процедурами, що описує ситуацію на складі (див. Розділ 2.2) і забезпечує коригування інформації при завезенні / вивезенні продукції. Передбачити, що склад має обмежений фіксований обсяг.

15. Вихідний список містить інформацію про студентів факультету інформатики та обчислювальної техніки. Сформувати АКО-ієрархію факультету, передбачивши автоматичну генерацію фреймів, що описують конкретного студента, на основі фрейму-прототипу, із завданням додаткової інформації про нього (ім'я, вік і т.ін.) в діалоговому режимі.

Приклад-пояснення: вид вхідного списку -

```
          (АВТФ
```

```
(A-1-20 (ІВАНЕНКО ПЕТРЕНКО .... ))
(A-2-20 (СИДОРЧУК ФРАНЧУК .... ))
          ................
(A-14-20 (ЄГОРІВ ПОПЕНКО....))   )
```

Додаток.

КОРОТКИЙ ОПИС ВИКОРИСТАНИХ ФУНКЦІЙ МОВИ ФРЛ

1. Позначення для функцій ФРЛ

У таблицях, наведених нижче, будемо використовувати наступні позначення для аргументів функцій в мові ФРЛ:

f - ім'я фрейму; ***fs*** - фрейм-структура;
s - ім'я слота; ***ss*** - слот;
a - ім'я аспекту; ***as*** - аспект;
v - ім'я даного; ***vs*** - дане;
l - ім'я коментаря (мітка); ***ls*** - коментар;
m - ім'я повідомлення; ***MS*** - повідомлення;
pnl - список імен процедур (procedure name list)
fnl - список імен фреймів (frame name list)
fvl - список даних ФРЛ (frame value list)
fn - ім'я функції (function name)

Зауваження: Факультативні аргументи ФРЛ-функцій слідують за обов'язковими і в даному описі відокремлюються двокрапкою. Альтернативні аргументи вказані в фігурних дужках.

2. Константи

Синтаксис	Семантика
**** AKO-INSTANCE ****	Список властив-й для зберігання імен інверсних відносин ***(CDR '* AKO-INSTANCE *) => ((AKO. INSTANCE) (INSTANCE. AKO))***
AKO	***AKO***
INSTANCE	***INSTANCE***
$VALUE	***$VALUE***
$DEFAULT	***$DEFAULT***
$IF-NEEDED	***$IF-NEEDED***
$IF-REMOVED	***$IF-REMOVED***
$IF-ADDED	***$IF-ADDED***
$IF-INSTANTIATED	***$IF-INSTANTIATED***
$REQUIRE	***$REQUIRE***
$IF-GETED	***$IF-GETED***
$IF-ADD	***$IF-ADD***
$IF-REM	***$IF-REM***
$IF-GET	***$IF-GET***

3. Стекі

Ім’я стеку	Призначення	Використовується функціями
FRAMES* (список властивостей)	Має імена активних фреймів	**DEFRAME, FRESET, FPRINT, FASSERT, FSAVE, DEFRAMEQ, FASSERTQ, FDESTROY**
PROCEDURES* (список властивостей)	Має імена актичних процедур	**PASSERT, PRESET, PSAVE, PASSERTQ**
FGENAMELIST* (змінна)	Має інформацію для генерації унікальних імен фреймів (реалізований як список властивостей)	**FGENAME, FGETNAME**

4. Глобальні змінні

Синтаксис	Семантика
FRAME*	Тіло фрейма зразу після ***FRAME***? або ***FNAME***?
FNAME*	Ім’я фрейма зразу після ***FRAME***? або ***FNAME***?
DESCR*	Тіло дескриптора зразу після ***DNAME***?
DNAME*	Ім’я дескриптора зразу після ***DNAME***?
MATCHED*	Список об’єктів, які були співставлені після ***FMATCH***?
UNMATCHED*	Список об’єктів, які не були співставлені після ***FMATCH***?

5. Зміна мережі фреймів і процедур

Синтаксис	Семантика
(PASSERT fn bd)	Визначається або переопределяется процедура з ім'ям **fn** і тілом **bd**. Ім'я процедури заноситься в стек ***PROCEDURES***. Результат - ім'я процедури.
(PASSERTQ fn bd)	Те ж що і **PASSERT**, але при зверненні аргументи не обчислюються.
(DEFRAME f : ss1 ... ssn)	Створюється новий фрейм, що містить зазначені слоти. Результат - ім'я створеного фрейму. Приєднані процедури не активізуються.
(DEFRAMEQ f : ss1 ... ssn)	Аналог **DEFRAME**, але аргументи **DEFRAMEQ** не вирахував при зверненні.
(FASSERT f : ss1 … ssn)	Створюється новий або поповнюється старий кадр. Результат - ім'я фрейму. Приєднані процедури виконуються.

(FASSERTQ f : ss1 … ssn)	Аналог **FASSERT**, але аргументи **FASSERTQ** не вирахував при зверненні.
(FRENAME f1 : f2)	Фрейм з ім'ям **f1** перейменовується у фрейм з ім'ям **f2**. Результат - нове ім'я.
(FNAME {f | fs})	Результат - ім'я фрейму. Якщо такий фрейм не знайдено - він створюється.
(FRAME {f | fs})	Результат - покажчик на фрейм-структуру. Якщо фрейм не знайдено - він створюється.
(FPUT- f : s a v l m)	У тіло фрейма **f**, додається слот **s**, аспект **a**, дане **v**, мітка **l**, коментар **m**. Якщо яка-небудь зі структур: **f, s, a, v, l, m** вже існувала, то в неї або додається нова інформація, або ця структура залишається без зміни. Якщо до моменту виконання **FPUT-** яка-небудь зі структур: **f, s, a, v, l, m** не існувала - вона створюється. Приєднані процедури не виконуються.
(FPUT-STRUCTURE- f) **(FPUT-STRUCTURE- f ss)** **(FPUT-STRUCTURE- f s as)** **(FPUT-STRUCTURE- f s a vs)** **(FPUT-STRUCTURE- f s a v ls)** **(FPUT-STRUCTURE- f s a v l ms)**	Те ж, що і в **FPUT-**, але останній аргумент звернення трактується як готова відповідна структура, що додається цілком під фрейм
(FPUT f : s a v l m)	У тіло фрейма **f**, додається слот **s**, аспект **a**, дане **v**, мітка **l**, коментар **m**. Якщо яка-небудь зі структур: **f, s, a, v, l, m** вже існувала, то в неї або додається нова інформація, або ця структура залишається без зміни. Якщо до моменту виконання **FPUT-** яка-небудь зі структур: **f, s, a, v, l, m** не існувала - вона створюється. Приєднані процедури виконуються.
(FPUTV f s v)	**=(FPUT f s $VALUE v)**
(FPUT-STRUCTURE f) **(FPUT-STRUCTURE f ss)** **(FPUT-STRUCTURE f s as)** **(FPUT-STRUCTURE f s a vs)** **(FPUT-STRUCTURE f s a v ls)** **(FPUT-STRUCTURE f s a v l ms)**	Те ж, що і в **FPUT**, але останній аргумент звернення трактується як готова відповідна структура, яка буде додана у фрейм.
(FPUT-STRUC f sb)	**=(FPUT-STRUCTURE f sb)**
(FPUT-STRUC f s ab)	**=(FPUT-STRUCTURE f s ab)**
(FPUT-STRUC f s a d)	**=(FPUT-STRUCTURE f s a d)**
(FPUT-STRUC f s a v c)	**=(FPUT-STRUCTURE f s a v c)**
(FPUT-STRUC f s a v l m)	**=(FPUT-STRUCTURE f s a v l m)**
(FDELETE f : s a v l m)	Те ж, що і **FREMOVE**, але приєднані процедури не активізуються і результат - покажчик на

	змінену подструктуру фрейма.
(FDEL-STRUCTURE f ss) **(FDEL-STRUCTURE f s as)** **(FDEL-STRUCTURE f s a vs)** **(FDEL-STRUCTURE f s a v ls)** **(FDEL-STRUCTURE f s a v l ms)**	Видаляє подструктуру фрейма **f**, що локалізуються іншими аргументами звернення. Результат - змінена підструктура або **NIL**, якщо видалення не відбулося. Приєднані процедури не виконуються. Число аргументів - від 1 до 6.
(FREMOVE f : s a v l m)	Видаляє з фрейма подструктуру, що локалізуються аргументами звернення. Останній аргумент звернення задає ім'я видаляється підструктури. Число аргументів - від 1 до 6. Результат - останній аргумент у зверненні. Приєднані процедури виконуються.
(FREM-STRUCTURE- f ss) **(FREM-STRUCTURE- f s as)** **(FREM-STRUCTURE- f s a vs)** **(FREM-STRUCTURE- f s a v ls)** **(FREM-STRUCTURE- f s a v l ms)**	Те ж що і **FDEL-STRUCTURE**, з тією різницею що приєднані процедури не виконуються
(FREM-STRUC f s a v l m)	**=(FREM-STRUCTURE f s a v l m)**
(FREP-STRUCTURE f ss) **(FREP-STRUCTURE f s as)** **(FREP-STRUCTURE f s a vs)** **(FREP-STRUCTURE f s a v ls)** **(FREP-STRUCTURE f s a v l ms)**	Останній аргумент звернення трактується як структура відповідного рівня. Вона повністю замінює у вхідному фреймі відповідну структуру. Результат - змінена підструктура. Число аргументів - від 1 до 5. Приєднані процедури виконуються.
(FREP-STRUCTURE- f ss) **(FREP-STRUCTURE- f s as)** **(FREP-STRUCTURE- f s a vs)** **(FREP-STRUCTURE- f s a v ls)** **(FREP-STRUCTURE- f s a v l ms)**	Останній аргумент звернення трактується як структура відповідного рівня. Вона повністю замінює у вхідному фреймі відповідну структуру. Результат - змінена підструктура. Число аргументів - від 1 до 5. Приєднані процедури не виконуються.
(FREP-STRUC f s a v l m)	**=(FREP-STRUCTURE f s a v l m)**
(FINSTANTIATE f1 : f2)	Створюється екземпляр фрейма **f1** з ім'ям **f2**. Якщо **f2** опущено, то створюється фрейм з системним ім'ям, в створюваний екземпляр поміщаються також значення, що виробляються процедурами, приєднаними до аспектам **$IF-INSTANTIATED** кожного слота фрейма **f**. Результат - ім'я створеного фрейму.
(FCLEAN f s : a)	Видаляє з фрейма **f** слота **s** аспекту a всі дані, які не задовольняють процедурам, успадковані з аспекту **$REQUIRE**. Якщо аспект a опущений, то **a = $VALUE**. Повертає список видалених даних.
(FINSTANCE f sb*)	Створює екземпляр фрейма **f** зі слотами **s1 s2 ... sN**.
(FREVADD s)	Створює зворотне посилання **s** з

	фрейма :**VALUE** на фрейм :**FRAME** (в аспекті :**FACET**).
(FREVREM s)	Видаляє зворотне посилання **s** з фрейма **:VALUE** на фрейм **:FRAME** (в аспекті **:FACET**).
(FORWADD s)	Створює пряме посилання **s** з фрейма **:VALUE** на фрейм **:FRAME** (в аспекті **:FACET**).
(FORWREM s)	Видаляє пряме посилання **s** з фрейма **:VALUE** на фрейм **:FRAME** (в аспекті **:FACET**).
(PRESET : pnl)	Зазначені процедури видаляються з системи. Результат список імен віддалених процедур. Якщо **pnl** в зверненні опущений, то вважається, що **pnl = *PROCEDURES***.
(FRESET : fnl)	Зазначені фрейми затираються в оперативній пам'яті. Приєднані процедури не виконуються. Результат - список імен затертих фреймів. Якщо **fnl** в зверненні опущений, то вважається, що **fnl = *FRAMES***.
(DEDESCR q sb*)	Створює новий дескриптор **q** зі слотами **s1 ... sN**.
(DINSTANCE q sb*)	Створює екземпляр дескриптора **q** зі слотами **s1 ... sN**

6. Вилучення інформації з мережі фреймів

Синтаксис	Семантика
(FPRINT : fnl)	Зазначені фрейми виводяться на екран у вигляді, зручному для сприйняття. За замовчуванням **fnl = *FRAMES***. Результат - список фреймів, які вдалося надрукувати.
(PPRINT : pnl)	Зазначені процедури виводяться на екран у вигляді, зручному для сприйняття. За замовчуванням **pnl = *PROCEDURES*** Результат - список процедур, які вдалося віддрукувати.
(FGET f : s a k)	Результат - список підструктур або їх імен, витягнутих з мережі фреймів на підставі наведених аргументів. Наприклад: **(FGET f)** - повертає список імен слотів фрейма **f**. **(FGET f s a)** - повертає список імен даних з аспекту **a** слота **s** фрейма **f**. За замовчуванням **a = $ VALUE** **k = (L -C! @ A H 1)**

	Приєднані процедури виконуються. Коментарі обробляються.
(FGET1 f : s a v l)	Те ж, що і **FGET** з тією різницею, що в якщо результат **FGET** дорівнює **NIL**, то запит поширюється на фрейми, імена яких містяться в якості імен даних слота **AKO** до тих пір, поки один з них не видасть результат, відмінний від **NIL**.
(FGETN f : s a v l)	Те ж, що і **FGET1**, але результуючий запит являє конкатенацію запитів до всіх можливих фрейми уздовж **AKO** зв'язку.
(FGET- f s a : v l)	Те ж, що і **FGET**, але не виконуються приєднані процедури і не обробляються коментарі. Принцип умовчання на аспект "**a**" не поширюється.
(FGET1- f s a : v l)	Те ж, що і **FGET1**, але не виконуються приєднані процедури і не обробляються коментарі. Принцип умовчання на аспект "**a**" не поширюється.
(FGETN- f s a : v l)	Те ж, що і **FGETN**, але не виконуються приєднані процедури і не обробляються коментарі. Принцип умовчання на аспект "**a**" не поширюється.
(FGET-STRUCTURE f :s a v l)	Те ж, що і **FGETN**, але не виконуються. Результат - список підструктур виділених з фрейм структури на підставі наведених аргументів. наприклад: **(FGET-STRUCTURE f s a)** - повертає аспект "**a**" Спорт в подробицях "**s**" фрейму "**f**". За замовчуванням **a = $ VALUE** Приєднані процедури виконуються. Коментарі обробляються.
(FGET-STRUCTURE1 f :s a v l)	Те ж, що і **FGET-STRUCTURE**, але в разі якщо результат **FGET-STRUCTURE nil** запит поширюється на фрейми, імена яких містяться в якості імен даних слота **AKO**, до тих пір поки один з них не видасть результат відмінний від nil.
(FGET-STRUCTUREN f :s a v l)	Те ж, що і **FGET-STRUCTURE1**, але результуючий запит являє конкатенацію запитів до всіх можливих фрейми уздовж **AKO** зв'язку.
(FGET-STRUCTURE- f :s a v l) **(FGET-STRUCTURE1- f :s a v l)** **(FGET-STRUCTUREN- f :s a v l)**	Те ж, що і **FGET-STRUCTURE**, **FGET-STRUCTURE1**, **FGET-STRUCTUREN**, але не виконуються приєднані процедури і не обробляються коментарі. Принцип умовчання на аспект "**a**" не поширюється.

(FGETE f s : a)	**=(FGET f s a '(E))**
(FGETV f s : k)	**= (FGET f s $ VALUE k)**, причому в разі неуспіху результат формують процедури, успадковані з аспекту **$IF-REQ** слота "**s**" фрейму "**f**".
(FGET-STRUC f : s a v l m)	**=(FGET-STRUCTURE f : s a v l m)**
(FGET-SEL f s a : k : l1 m1 ... lN mN)	Витягує (за допомогою **FGET** з урахуванням ключів **k**) з аспекту **a** слота **s** фрейма **f**, всі дані, які містять коментарі **(l1 m1) ... (lN mN)**.
(FGETIND f1 (s+) : a k)	Ланцюговий варіант **FGET**. Повертає **(FGET fN sN: a k)**, де **f2 = (FGET f1 s1: a), f3 = (FGET f2 s2: a)** і т.д. Якщо **f2** або **f3** або ... або **fN** є багатоелементними списками, то результатом **FGET** від такого списку буде конкатенація результатів **FGETi**в від елементів цього списку.
(FHERITAGE f : s)	Повертає список імен слотів фрейма **f** і всіх фреймів, пов'язаних з ним через слот **s**. Якщо слот **s** не заданий, то **s = AKO**.
(FDESCENDANTS f : s)	Повертає список всіх імен фреймів (за винятком самого фрейму **f**) дерева відносини s з коренем в **f**. Якщо **s** не задано, то **s = AKO**.
(FTREE f : s)	Повертає спискового подання дерева відносини **s** з коренем в **f.** Якщо **s** не задано, то **s = AKO**.
(FRINGE f : s)	Повертає список всіх листя дерева відносини **s** з коренем в **f**. Якщо **s** не задано, то **s = AKO**.
(FCHILDREN f : s)	Повертає список всіх безпосередніх нащадків фрейма **f** в дереві відносини **s**. Якщо **s** не задано, то **s = AKO**.
(FINHERIT f s : a)	**=(FGET f s a (C 0))**
(FINHERIT1 f s : a)	**=(FGET f s a (C 0))**
(FINHERIT2 f s : a)	**=(FGET f s a (C 0 2))**
(FSLOTS f)	Результат - список імен слотів фрейма **f**
(FRAMES)	Повертає в якості результату список фреймів, що знаходяться в оперативній пам'яті (активних фреймів).
(PROCEDURES)	Повертає в якості результату список процедур, які перебувають в оперативній пам'яті (активних фреймів).
(DESCRS)	Список всіх дескрипторів.
(FQUERY f s a : ref field relation)	Пошук в мережі фреймів.

7. Предикати

Синтаксис	Семантика
(FACTIV? f)	Результат - **T**, якщо фрейм **f** знаходиться в оперативній пам'яті. Інакше - **nil**.
(PACTIV? pn)	Результат - **T**, якщо процедура **pn** знаходиться в оперативній пам'яті. Інакше - **nil**.
(FRAME? f)	Результат - фрейм-структура фрейму **f**, якщо цей фрейм знаходиться в оперативній пам'яті або відкритому розділі ВБО. Інакше - **nil**.
(DNAME? q)	Ім'я дескриптора **q**, якщо **q** є дескриптором в оперативній пам'яті або в базі об'єктів (в цьому випадку він завантажується в оперативну пам'ять). **(CAR q)**, якщо **q** - список. В іншому випадку **NIL**.
(AKO? f1 fN)	Повертає список імен фреймів виду **(f1 ... fN)**, де **f2** є **AKO**-прототипом для **f1**, **f3** є **AKO**-прототипом для **f2** і т.д. Якщо такий список побудувати не вдається, то результат **NIL**.
(INSTANCE? f1 fN)	Аналогічно **AKO?**, але замість **AKO**-прототипів беруться **INSTANCE**-екземпляри.
(FLINK? f1 fN : s)	Повертає список імен фреймів виду **(f1 ... fN)**, де **f1** безпосередньо пов'язаний зв'язком **s** з **f2**, **f2** безпосередньо пов'язаний зв'язком **s** з **f3** і т.д. За замовчуванням **s = AKO**. Якщо такий список побудувати не вдається, то результат **NIL**.
(ISA? f1 fn : s)	Те ж, що і **FLINK?**, але зв'язку **s** можуть успадковуватися.
(TOPIC? f1 f2 : s)	Якщо існує такий фрейм **fx**, що **(FLINK? F1 fx s) & (FLINK? F2 fx s)**, то повертає **(FLINK? F2 fx s).**
(FCOMMENT? d l m*)	Повертає дане **d**, якщо воно містить коментар з міткою l і повідомленнями **m***.
(FCHECK? f s : d)	Повертає **T**, якщо дане **d** задовольняє умовам, які успадковуються з фрейма **f**, слота **s**, аспекту **$REQUIRE**. За замовчуванням береться дане з фрейма **f**, слота **s**, аспекту **$VALUE**.
(FMATCH? f q)	Повертає **T**, якщо фрейм **f** можна порівняти з дескриптором **q**. Формуються також глобальні змінні ***MATCHED*** і ***UNMATCHED***

8. Інтерфейс з віртуальною базою об’єктів

Синтаксис	Семантика
(OPEN-BASE : bn)	Завантажується ВБО **bn** з директорії з ім'ям **bn**.

	Результат - ім'я завантаженої ВБО. Якщо ВБО із зазначеним ім'ям немає в поточній директорії, то вона може бути створена.
(CLOSE-BASE)	Закриває ВБО. Результат - ім'я ВБО, з якої припинено роботу.
(FOPEN name)	Розділ "**name**" ВБО оголошується відкритим. Розділ, який був раніше відкритим, оголошується пасивним. Результат - список з "**name**". Якщо розділ відкрити не вдалося (ВБО була попередньо завантажена), то **NIL**. Якщо "**name**" опущено в зверненні, то відкривається глобальний розділ ВБО з ім'ям **GLB**. В цьому випадку результат - **NIL**.
(FCLOSE)	Поточний відкритий розділ ВБО закривається. Активним стає раніше пасивний розділ. Результат - список, з імені закритого розділу.
(FLOAD{f | fnl | pn | pnl)	Зазначені фрейми або процедури завантажуються в оперативну пам'ять з поточного відкритого розділу ВБО. *Увага!!!* Якщо об'єкт не знайдено у відкритому розділі ВБО, проводиться спроба завантажити його з глобального розділу ВБО. Результат - ім'я завантаженого об'єкта або **NIL**, якщо об'єкт в ВБО не найден.
(FSAVE : fnl)	Зазначені фрейми поміщаються у відкритий розділ ВБО, після чого вони видаляються з оперативної пам'яті. Результат - список імен збережених об'єктів. Якщо **fnl** опущений, то **fnl = *FRAMES***.
(PSAVE : pnl)	Те ж, що і **FSAVE**, але в разі, коли **fnl** опущений, Вважається, що **fnl=*PROCEDURES*.**
(FPSAVE)	**= (FSAVE) + (PSAVE)**
(MAINT kw : name1 name2 flag)	Сервісна процедура взаємодії з ВБО. **kw** задає режим роботи функції **MAINT**. **kw = DIR** - друк змісту розділу **name1** ВБО, якщо **name1** опущено - друк змісту всієї ВБО. Якщо **kw = DELP - MAINT** видаляє розділ **name1** з ВБО. При kw = DELO - відбувається видалення об'єкта **name1** з розділу **name2**. Якщо **flag** не дорівнює **NIL**, то зміст не друкувати, а видається у вигляді результату.
(OGLAV : name flag)	Друк змісту розділу **name** ВБО, якщо name опущено - друк змісту всієї ВБО. Якщо **flag** не дорівнює **NIL**, то зміст не друкувати, а видається у вигляді результату.

(FDESTROY f)	Зазначений фрейм затирається в оперативній пам'яті і видаляється з ВБО. Результат - ім'я віддаленого фрейма.
(PDESTROY pn)	Зазначена процедура видаляється з системи. Результат ім'я процедури. Процедура видаляється також з ВБО.
(BASE_GC)	Процедура ущільнення інформації у відкритому розділі ВБО. В процесі своєї роботи друкує об'єм ВБО в байтах до і після ущільнення. Ця процедура запускається автоматично кожного разу при перевищенні обмежувача на розмір ВБО, що встановлюється функцією **BD_SIZE**.
(BD_SIZE : n)	Встановлює максимальний розмір в байтах розділу ВБО. Як тільки розмір розділу ВБО перевищить встановлену межу - відбувається його ущільнення. Якщо n опущено - як результат повертається значення поточної межі.

9. Активація процедур

Синтаксис	Семантика
(FAPPLY d)	Обчислює значення даного **d** з урахуванням коментарів з мітками **STATUS:**, **PARM:** і **PARMQ:**. Повертається отримане значення, укладену в дужки.
(FPROG (d*) (m*))	Послідовно обчислює (за допомогою **FAPPLY**) дані з (**d***) до тих пір, поки не буде отримано результат, відмінний від **NIL**, який і буде результатом **FPROG**. Якщо список (**m***) не пустили, то обчислюються тільки ті дані, які містять в коментарі з міткою **TYPE:** повідомлення з **(m *)**.
(FPROGL (d*))	Обчислює (за допомогою **FAPPLY**) дані з **(d*)**, враховуючи при цьому функтори **OR, NOT, ALT, IF, THEN, ELSE, FI.**
(FEXEC f s a (m*))	**=(FPROG (FGET f s a) (m*))**
(FNEED f s (m*))	**=(FEXEC f s $IF-NEEDED (m*))**
(FEVAL d : f s a v)	**= (FAPPLY d)**, причому створюється середовище, в якій: **:FRAME = f**, **:SLOT = s**, **:FACET = a**, **:VALUE = v**.
(FCHECK f s : v*)	Повертає список з результатів перевірок для кожного значення **v** з **(v *)**. Кожен результат перевірки має вигляд: **(<підсумок> <TRUE-список> <FALSE-список> <? - список>)** де **<підсумок> = T**, якщо **v** задовольняє всім умовам з **(FGET f s $REQUIRE (C))**; **<Підсумок> = NIL**, якщо є незадоволені умови; **<Підсумок> =?**, Якщо є невизначені умови. **<TRUE-список>** - це список всіх задовольнити умов і т.д.

	За замовчуванням **(v *) = (FGET f s $ VALUE (-H))**.
(FCHECK-SUMMARY f s v*)	Те ж, що і **FCHECK**, але повертає список підсумків.
(FCHECK-INST f)	Виконує для всіх значень всіх слотів фрейма **f** відповідні приєднані процедури, успадковані з аспекту **$IF-ADDED**.
(FGNEED f s : a k (m*))	Збігається з результатом **(FGET f s a k)**, якщо це не **NIL**. В іншому випадку збігається з результатом **(FNEED f s: (m *))**.
(FGNEEDR f s : a k (m*))	Те ж, що і **FGNEED**, але якщо спрацьовує **FNEED**, то її результат заноситься в аспект **a** слота **s** фрейма **f**.
(FTRACE f list)	Підключити до роботи ФРЛ-процесора процедури трасування. **f** - ім'я фрейму, що містить трасуючі процедури, а **list** - список виду **((<мета-ім'я-підструктури> <умова> ...)**. **<Мета-ім'я-підструктури>** - це є **FRAME, SLOT, VALUE, LABEL** або **MESSAGE** . **<Умова>** представляє один з наступних атомів: **IF-ADDED, IF-REMOVED, IF-GETED**, що інтерпретується наступним чином: **IF-ADDED** - трасуючі процедури запускаються при додаванні в систему структури з відповідним мета-ім'ям; **IF-REMOVED** - трасуючі процедури запускаються при видаленні з системи структури з відповідним мета-ім'ям; **IF-GETED** - трасуючі процедури запускаються при вилученні з мережі структури з відповідним мета-ім'ям. Якщо **list** опущений, то підключаються всі можливі трасуючі процедури фрейма **f**.
(FUNTRACE)	Відключає всі трасуючі процедури.

10. Допоміжні функції

Синтаксис	Семантика
(SWITCH-AKO : s)	Спадкування прямує уздовж слота **s**. За замовчуванням **s = AKO**.
(SWITCH-INSTANCE : s)	Встановлює ім'я зв'язку, реверсивної по відношенню до **AKO**. За замовчуванням **s = INSTANCE**.
(FREF d : k)	Визначає референта для даного **d** відповідно до ключами **k**.
(FINDIRECT d : k)	Визначає референта для непрямого даного **d**.
(FINDCOMMENT (d*) l m)	Повертає **NIL**, якщо жодна дане з **(d*)** не має коментаря з міткою **l** і повідомленням **m**.
(FADD-COMMENT d c)	Вбудовує коментар **c** в дане **d**.
(REMDUP (n*))	Повертає **(n*)** без повторень.

(FMINUS list1 list2)	Результат - список **list1** з якого видалені всі елементи, що входять до **list2**.
(ADJOIN obj list)	Аналогічно **CONS** з тією різницею, що **ADJ** не додавати в список **list** повторювані елементи
(ADD_TO_FULL list1 list2)	Аналогічно **APPEND** з тією різницею, що результуючий список створюється без повторюваних елементів.
(FGENAME : name)	Результат - унікальне системне ім'я, що має в якості префікса значення **name**. За замовчуванням **name = SYS**.
(FGETNAME : name)	Результат - остання сгенерированное унікальне системне ім'я, що має в якості префікса значення **name.** За замовчуванням **name = SYS**.
(EVLIST list)	Результат - список, що складається з результатів обчислення всіх відповідних елементів **list**.

ЛІТЕРАТУРА

1. Artificial Intelligence: A Modern Approach, Fourth Edition / Ed. Stuart Russell and Peter Norvig, - Prentice Hall: 2020. - ISBN 9780134610993. - 1115 p.
2. Полыскалин В.Я., Баклан И.В. и др. Информационное обеспечение интегрированных АСУ ГПС. Том 2. - М.: Машиностроение, 1989. - 888 с.
3. Баклан І.В. Експертні системи. Курс лекцій /Навчальний посібник. - К.: НАУ, 2012. - 132 с.
4. Уэно Х., Кояма Т. и др. Представление и использование знаний. - М.: Мир, 1989, 220 стр.
5. Семенова Е.Т. Язык программирования ЛИСП 1.5 - М.: МЭИ, 1977.
6. Байдун В.В., Кружилов С.И. и др., Программирование на языке ЛИСП в системе muLISP-90 - М.: МЭИ, 1993.
7. Минский М. Фреймы для представления знаний. - М.: Энергия, 1979, 152 стр.
8. Семенова Е.Т. Представление знаний в системе LISP/FRL. -М.: МЭИ, 1987, 104 стр.
9. Байдун В.В., Бунин А.И. Интегрированная инструментальная среда для разработки экспертных систем // Моделирование и искуственный интеллект, - М.: МИРЭА, 1988.
10. Языки и системы представления знаний (язык программирования ФРЛ). Байдун В.В., Бунин А.И., Чернов П.Л. ─ М.: Моск. энерг. ин-т, 1993. ─ 44 с.