

Міністерство освіти і науки України  
Львівський національний університет імені Івана Франка  
*Кафедра оптоелектроніки та інформаційних  
технологій*

Звіт  
про виконання лабораторної роботи №3  
З курсу “Системи штучного інтелекту”  
на тему:  
« РОБОТА З ФАКТАМИ В CLIPS »

**Виконав:**  
**Студент групи ФеС-21**  
**Шавало А.А.**  
**Перевірив:**  
**Грабовський В.А.**

Львів - 2024

**Мета роботи:** Отримати навички введення в CLIPS впорядкованих та неупорядкованих фактів та їх перетворень.

**Завдання до роботи:**

Вивчити роль та значення фактів у експертних системах. Засвоїти особливості різних методів представлення і введення впорядкованих і неупорядкованих фактів у середовищі CLIPS та дії функцій, призначених для маніпуляції з введеними фактами – перевірки наявності введених фактів у робочій пам'яті, їх модифікації і зміни, видалення, очищення списку фактів, глибокого очищення середовища. Засвоїти особливості введення одиночних фактів та їх масивів і команд, які використовуються при цьому.

***Обладнання та програмне забезпечення:***

– Середовище програмування CLIPS.

***Теоретичні відомості***

Робота експертної системи, за визначенням, ґрунтується на використанні закладених в неї знань, і тому її функціонування неможливе без наявності в ній спеціально створеної бази знань. Найчастіше ці знання в експертних системах представляються набором певних фактів і правил, які задаються з використанням деякої мови опису знань – такі системи ще називаються продукційними. Тому будь-яка експертна система повинна володіти арсеналом засобів для створення, введення фактів і правил, а також маніпуляцій ними.

## Факти

Для успішного рішення експертною системою, створеною в середовищі CLIPS, поставленої задачі їй повинна бути надана інформація, на підставі якої вона могла б здійснювати свої міркування і формувати висновок. Факти і є тими «фрагментами» інформації, які використовуються в мові CLIPS.

Факти є однією з основних форм представлення інформації в системі CLIPS, складаються з символного поля імені-відношення, за яким слідує або нелімітована кількість символних полів (розділених недрукованими символами, зазвичай – пробілами) (у випадку впорядкованих фактів), або слотів і пов'язаних з ними значень (у неупорядкованих фактах), та зберігаються в оперативній пам'яті комп'ютера. Весь факт (як і кожен слот) обмежується відкриваючою і закриваючою круглими дужками.

Для переглядання поточного списку фактів використовується команда *facts*.

Як уже згадувалося, CLIPS підтримує два формати представлення фактів: врегульовані (або впорядковані) і неврегульовані (або неупорядковані, чи шаблонні) факти. Практично, будь-який фрагмент інформації можна представити у вигляді як впорядкованого, так і неупорядкованого факту. Наприклад, той факт, що студент Іван Іванович Іванів, віком 21 рік, має голубі очі і чорне волосся, може бути представлений у вигляді як впорядкованого факту:

(student Ivan Ivanovych Ivaniv aged 21 has blue eyes and black hair),  
так і неупорядкованого:

```
(student  
  (name«Ivan Ivanovych Ivaniv»)  
  (age 21)  
  (eye-color blue)  
  (hair-color black)).
```

### 3.1.1. Впорядковані факти.

Впорядковані факти складаються з виразу символьного типу, за яким слідує необмежена послідовність (можливо, і порожня) з полів, розділених недрукованими символами (пропусками). Перше поле запису визначає «ім'я відношення» (або «зв'язок» факту), яке застосовується до полів, що залишилися. Весь запис поміщається в круглі дужки.

За винятком першого поля, що обов'язково повинне бути типу *symbol*, інші поля у факті можуть зберігати дані будь-якого примітивного типу CLIPS. Потрібно також пам'ятати, що слова: *test*, *and*, *or*, *not*, *declare*, *logical*, *object*, *exist* й *forall* – зарезервовані й не можуть бути використані як перше поле впорядкованого факту.

Синтаксис впорядкованого факту:

(дані\_типу\_symbol [поле]\*).

Приклади впорядкованих фактів:

(student Sergiy Petrenko)

(classmates Ivaniv Petriv Sydorenko)

(color red).

### Невпорядковані факти.

На відміну від впорядкованих, неупорядковані (шаблонні) факти дають можливість користувачеві задавати певну абстрактну структуру факту, задаючи ім'я на кожному з полів (слотів) факту з певним іменем-відношенням. Це означає, що для забезпечення можливості створення конкретного факту інтерпретатору CLIPS необхідно попередньо повідомити інформацію про те, яким є список допустимих слотів для факту з вказаним іменем, тобто створити шаблон факту. Причому, як уже згадувалося, порядок слотів у неупорядкованому факті не важливий. Визначення неупорядкованого факту, як і впорядкованого, обмежується круглими дужками.

### Конструктор *deftemplate*.

Для задання шаблону, який потім може використовуватися і при доступі до полів (слотів) неупорядкованого факту за їх іменами, використовується конструктор *deftemplate*. Його створення і введення в CLIPS приводить до

появи в поточній базі знань системи інформації про шаблон факту, за допомогою якого в неї надалі можна буде додавати факти, що відповідають заданому шаблону.

Синтаксис конструктора *deftemplate*:

(deftemplate >[ “Необов’язковий коментар” ] <slot-definition>

...

<slot-definition>)

Імена шаблонів і слотів повинні бути значеннями типу *symbol*; на імена шаблонів, як і у випадку впорядкованих фактів, також поширюється заборона на використання слів, зарезервованих системою.

Коментарі, зазвичай, описують призначення шаблону; щоб вони зберігалися в базі знань системи і були доступні при перегляді визначення шаблону, їх потрібно помістити в лапки. Крім даного типу коментарів, у конструкторі *deftemplate* також застосовні звичайні коментарі CLIPS, що починаються із символу ; – такі коментарі повністю ігноруються системою CLIPS.

Опис синтаксису визначається таким чином:

::=(slot ) | (multislot ).

Відповідно, слот може бути простим або складеним (мультислотом).

Прості слоти факту, задані за допомогою ключового слова *slot* у відповідних конструкціях *deftemplate*, містять тільки одне значення примітивного типу CLIPS (такі слоти ще називають *однозначними слотами*). У простому слоті може бути збережене лише одне значення.

Складені слоти факту, задані за допомогою ключового слова *multislot*, можуть містити будь-яку кількість (від нуля і більше) значень декількох

примітивних ти-пів CLIPS – такі слоти ще називають *багатозначними слотами*; у складеному слоті факту може зберігатися список.

Отже, >в простому випадку складається з: ключового слова *slot* або *multislot*, що визначає тип слота; імені слота, яке є значенням типу *symbol*. необов'язкового обмеження на тип значення, що зберігається в слоті.

### **Команда *facts*.**

Перевірити поточний стан списку фактів можна або за допомогою введення діалоговому вікні команди *facts* (тоді список усіх наявних в системі фактів буде виведений в діалоговому вікні), або викликавши пункт 1 Facts Window меню

Window (тоді список наявних в системі фактів буде виведений у вікні Facts (MAIN), яке з'являється в полі дисплея; рис. 1).

Якщо кількість фактів, що містяться в списку, є великою, то іноді виникає необхідність мати можливість розглядати тільки потрібну на даний момент його частину. Така можливість забезпечується шляхом задання додаткових параметрів команди *facts*. Загалом, повний синтаксис команди *facts* такий:

(facts [ < start > [< end > [< maximum >]]])

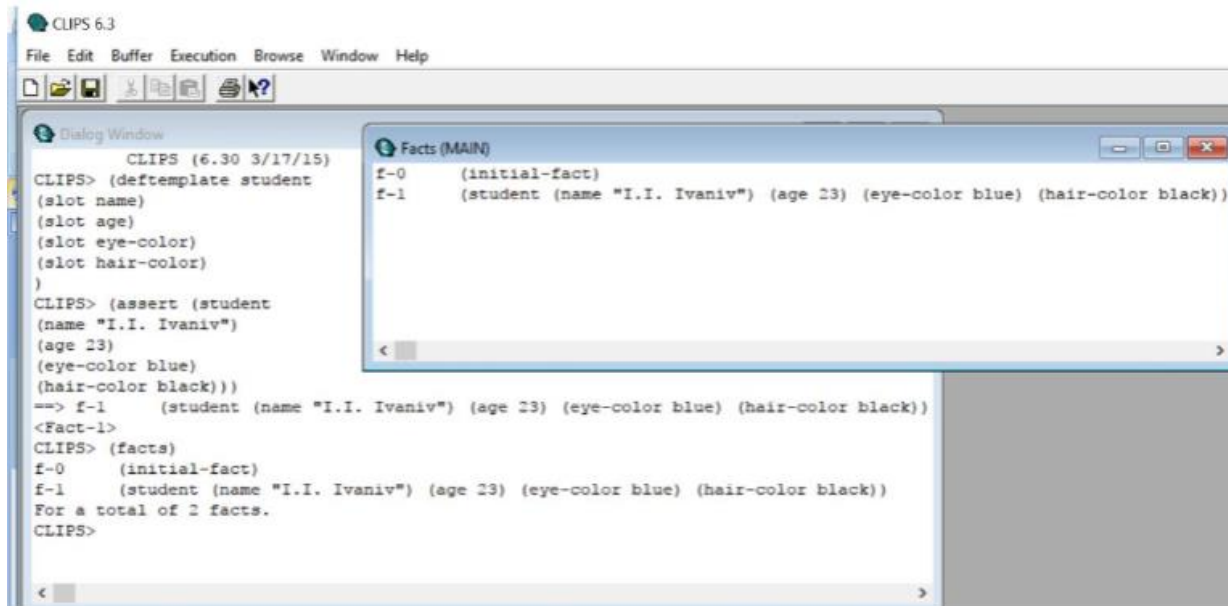


Рис. 1. Вигляд діалогового вікна з поточним списком фактів після введення ко-манди (facts) та вікна із списком фактів Facts (MAIN).

### Функція *assert*.

Нові факти можуть бути додані до наявних у списку з використанням ко-манди *assert*, яка має такий синтаксис:

(assert <fact>+)

Параметрами функції є послідовність фактів, які підлягають додаванню в список фактів; кількість фактів, що додаються, довільна.

Розглянемо приклад використання функції *assert* для введення впорядкова-ного факту (приклад введення неупорядкованого факту показано в п. 1.2). Для цього перейдемо в діалогове вікно CLIPS і введемо:

(assert (weather is fine))

При введенні команд необхідно чітко слідувати їх синтаксису, зокрема – бу-ти уважними з дужками. У тому випадку, якщо команда введена правильно, у список фактів додається факт (weather is fine) і діалоговому вікні CLIPS з'явиться результат її виконання: ==> f-1 (weather is fine)

що означає, що факт доданий в систему і отримав номер 1. При успішному виконанні функція *assert* повертає адресу останнього доданого факту.

При додаванні факту можна використовувати і виклики функцій; наприклад, виконання команди введення факту

(assert (totalcost (\* 10 13)))

викличе функцію множення і її результат запишеться як поле факту. При цьому в пам'ять додається факт (totalcost 130) і в діалоговому вікні з'явиться повідомлення:

CLIPS> (assert (totalcost (\* 10 13)))

==> f-1 (totalcost 130)

Якщо при додаванні деякого факту була допущена помилка (напр., повторне введення вже існуючого факту), команда припиняє роботу і повертає значення FALSE; якщо при цьому має місце синтаксична помилка, в діалоговому вікні виводиться відповідне повідомлення. Результати неправильного введення фактів відображаються у діалоговому вікні CLIPS і для випадку вказаного вище повторного введення факту мають вигляд:

CLIPS> (assert (totalcost (\* 10 13)))

==> f-1 (totalcost 130)

CLIPS> (assert (totalcost (\* 10 13)))

FALSE

CLIPS



### **Функція *fact-relation*.**

Функція *fact-relation* з синтаксисом

(fact-relation < fact-index >)

дозволяє визначити відношення (зв'язок) факту з вказаним індексом (або адре-сою) із шаблоном, заданим явно чи неявно, за допомогою якого створений факт. Цей зв'язок визначається за першим полем факту, яке завжди є простим полем і використовується CLIPS як ім'я шаблона, з яким зв'язаний факт. Функція повер-тає значення першого поля факту (ім'я факту), якщо даний факт існує, або зна-чення FALSE – якщо не існує.

### **Функція *fact-existp*.**

Для визначення, чи міститься в поточному списку фактів факт з вказаним індексом, використовується функція *fact-existp* з синтаксисом:

(fact-existp <fact-index>).

У випадку, якщо такий факт присутній у поточному списку фактів, то функція *fact-existp* повертає значення TRUE, якщо ні – FALSE.

Наприклад, послідовне виконання CLIPS команд:

(clear)

(assert-string «(Weather is fine)»)

(fact-existp 1)

(retract 1)

(fact-existp 1)

### Команда *save-facts*.

Команда збереження списку фактів у файл *save-facts* має синтаксис:

(save-facts <file-name> [visible | local \* <deftemplate-names>\*]) зберігає факти з поточного списку фактів у текстовий файл з вказаним іменем. функції існує можливість обмежити область фактів, що зберігаються. Для цього служить аргумент <межі-видимості>, який може приймати значення local

або visible. У випадку, якщо цей аргумент приймає значення visible, то зберігаються всі факти, які є у системі на момент запису; якщо ж аргумент – ключове слово local, то зберігаються тільки факти з поточного модуля. За замовчуванням аргумент <межі-видимості> приймає значення local. Після аргументу <межі-видимості> може бути представлений список визначених у системі шаблонів. У цьому випадку будуть збережені тільки ті факти, які створені із використанням зазначених шаблонів.

### Команда *load-facts*.

Для завантаження збережених раніше файлів використається команда *load-facts* з синтаксисом:

(load-facts <file-name> )

яка додає в поточний модуль факти, що записані у збереженому раніше файлі . У разі успіху повертає символ TRUE; в іншому – FALSE і відповідне повідомлення про помилку.

При завантаженні фактів командою *load-facts* необхідно пам'ятати, що якщо у записаному файлі є факти, пов'язані з явно створеними за допомогою конструктора *deftemplate* шаблонами, то на момент завантаження всі необхідні шаблони повинні бути вже визначені в системі. Якщо ця умова не буде виконана, то завантаження фактів закінчиться невдачею.

### 3.3. Конструктор *deffacts*.

Факти можна включати в базу даних не поодинці, а цілим масивом. Таке введення зручне, наприклад, при введенні фактів, що представляють початкові знання, або множини фактів, введення яких вимагає набору однотипних команд. Для цього в CLIPS призначений конструктор *deffacts*, синтаксис якого такий:

```
(deffacts <deffact name> [<optional comment>] < facts > *)
```

Услід за ключовим словом *deffacts* знаходиться обов'язкове ім'я цієї конструкції. Як ім'я може використовуватися будь-який допустимий вираз типу *symbol*, який служить більш як інформативний атрибут, ніж як ідентифікатор. За ім'ям слідує необов'язковий коментар, який зберігається у визначенні конструкції *deffacts* після завантаження відповідного визначення системою CLIPS. Услід за ім'ям або коментарем знаходяться факти, які вводяться в список фактів за допомогою команди *deffacts*, кількість яких загалом не обмежується.

Наприклад:

```
(deffacts student_list ; Список студентів
```

```
(student Ivaniv Ivan)
```

```
(student Petriv Petro) )
```

Вигляд результатів проведення вказаних операцій показано на рис. 2.

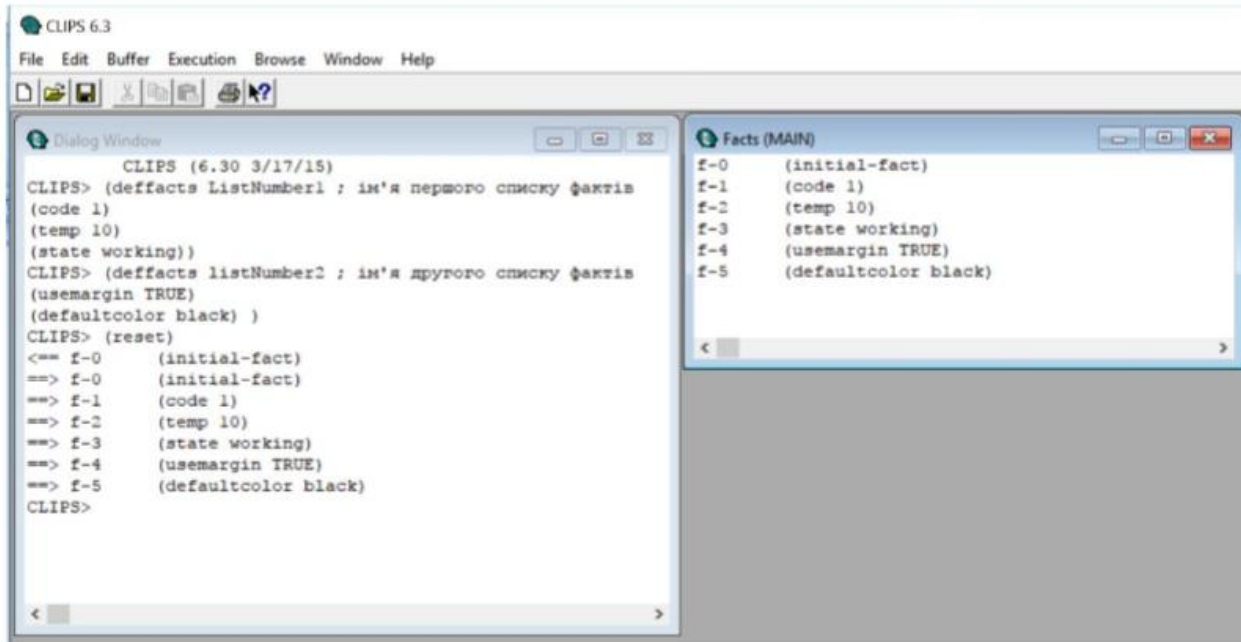


Рис. 2.Вигляд вікна CLIPS після введення списків фактів та виконання команди *reset* .

**Функція *duplicate*.**

Команда *duplicate* діє за таким же принципом, що і команда *modify* – за винятком того, що видалення первинного факту тут не відбувається – і дозволяє копіювати існуючі факти за заданим шаблоном, замінюючи вказані користувачем значення слотів.

Синтаксис команди:

(duplicate <fact-index><slot-modifer>+),

де після визначення факту (його індексу) вводиться список з одного або більше нових значень слотів (+) зазначеного шаблона.

Фактично, функція *duplicate* створює новий (з новим індексом) невпорядкований факт заданого шаблону шляхом копіювання в нього у незміненому вигляді групи слотів уже існуючого факту та модифікуючи значення вказаного в команді слота (слотів).

Результат виконання команди (duplicate 1) у випадку заборони і дозволу існування однакових фактів у базі знань показаний на рис. 3.

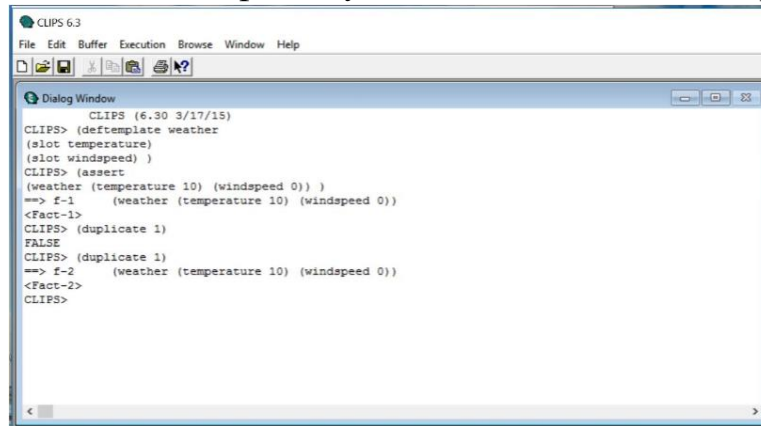


Рис. 3. Результат виконання CLIPS команди (duplicate 1) у випадку заборони і дозволу існування однакових фактів у базі знань.

### 3.5. Використання неупорядкованих фактів (шаблонів).

Як було відмічено вище, шаблон факту складається з імені факту і визначення полів для зберігання даних, які називаються слотами. Шаблони корисно використовувати для опису будь-якого об'єкту, що має набір атрибутів. Наприклад, якщо перед нами стоїть мета внести в систему інформацію про автомобіль, то його слотами можуть в найпростішому випадку бути колір і марка.

Розглянемо приклад шаблону для фактів, що описують автомобіль. Перейдемо у діалогове вікно CLIPS. Введемо команду *clear* для очищення середовища і введемо відповідний шаблон *deftemplate*:

(deftemplate car “Шаблон для опису автомобілів (slot color) (slot model)(multislot owner) )(Рис 4)

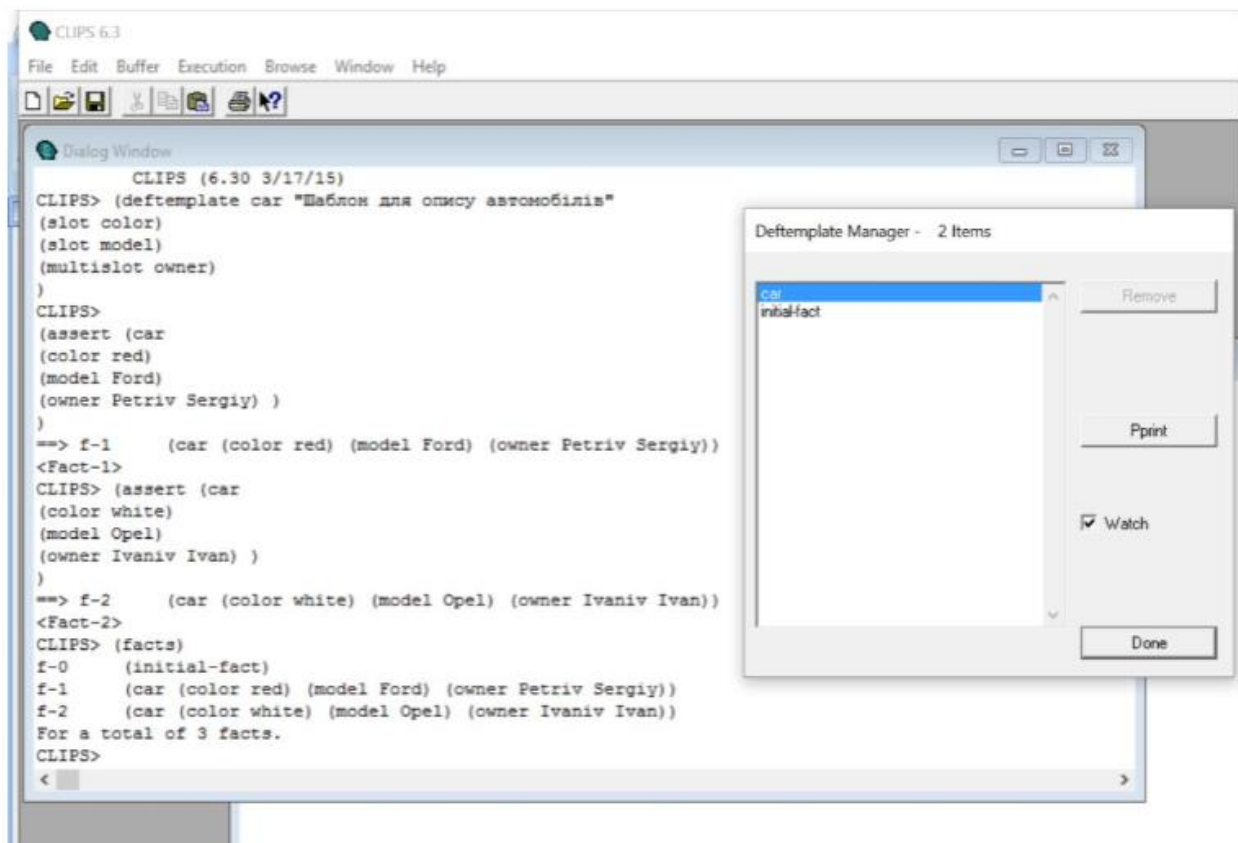


Рис. 4. Результати введення створеного шаблону car і двох фактів у діалоговому вікні і вікні менеджера шаблонів

### 3.6. Особливості застосування команд *modify*, *reset*, *clear*.

Існує множина додаткових функцій для роботи з фактами (найбільш поширені функції CLIPS можна побачити в Додатку 1). Їх опис і приклади використання можна знайти в on-line документації, а також в деяких з рекомендованих нижче літературних джерел.

Найчастіше під час роботи використовуються команди *modify*, *reset*, *clear*. Для зміни та дублювання шаблонних фактів служать команди *modify*,

*duplicate*. Параметрами цих команд (див. 4) є визначення факту (змінна – адреса або індекс факту), а також нові значення певних, вказаних користувачем, слотів.

Потрібно пам'ятати, що:

- при виконанні команди *modify* індекс модифікованого факту змінюється, а факт, який модифікується, видаляється зі списку фактів;
- при виконанні команди *duplicate* до списку існуючих фактів додається новий факт з новим індексом;
- команда *reset* очищає список фактів, а потім додає в нього факти, оголошені конструкторами *deffacts*, включаючи факт f-0 (*initial-fact*); зазвичай використовується у поєднанні з командою *run* для перезапуску написаної програми;
- на відміну від команди *reset*, команда *clear* виконує повне очищення виконуваного середовища і не додає у пам'ять системи ніяких фактів, окрім *initial-fact*; при цьому система очищається не тільки фактів, але також і від всіх списків, правил, змінних, шаблонів, які були в неї введені програмно або користувачем до моменту застосування команди *clear*.



## Хід роботи

1. Ознайомився з особливостями опису, створення та введення впорядкованих і неупорядкованих фактів у CLIPS та маніпуляцій з ними.
2. Створив список впорядкованих фактів у вигляді списку студентів групи у вигляді (student <name><date of birth><kurs>) на рис. 5.

```
CLIPS> (student "Ivanov" "2000-05-15" 2)
CLIPS> (student "Petrov" "1999-08-10" 3)
```

Рис. 5. Створення впорядкованих фактів

3. Використовуючи команду *assert*, створив список впорядкованих фактів у CLIPS, додаючи їх по черзі в діалоговому вікні (див. Рис. 6).
- ```
CLIPS> (assert (student "Golchuk" "2000-09-12" 4))
==> f-1      (student "Golchuk" "2000-09-12" 4)
<Fact-1>
CLIPS> (assert (student "Mihalchuk" "2001-05-20" 2))
==> f-2      (student "Mihalchuk" "2001-05-20" 2)
<Fact-2>
CLIPS>
```

Рис. 6. Створив список впорядкований список за допомогою assert

4. Переглянув введені факти, використовуючи команду *facts* (див. Рис. 7).

```
CLIPS> (facts)
f-0      (initial-fact)
f-1      (student "Golchuk" "2000-09-12" 4)
f-2      (student "Mihalchuk" "2001-05-20" 2)
For a total of 3 facts.
CLIPS> ■
```

Рис. 7.

5. Переглянув список введених фактів, використовуючи “Deftemplate Manager” з меню “Browse” (див Рис. 8).

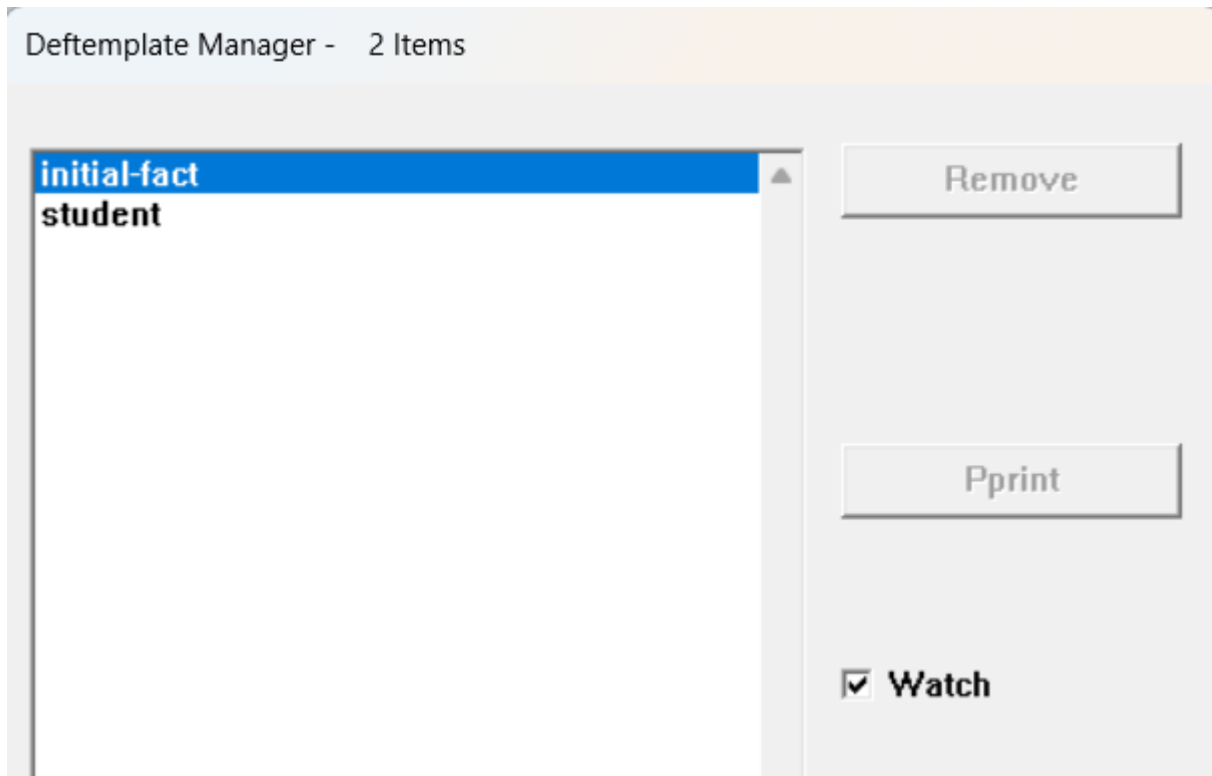


Рис. 8.

6. Переглянув список введених фактів, використовуючи опцію 1 Facts Windows з меню Windows

(див Рис. 9).

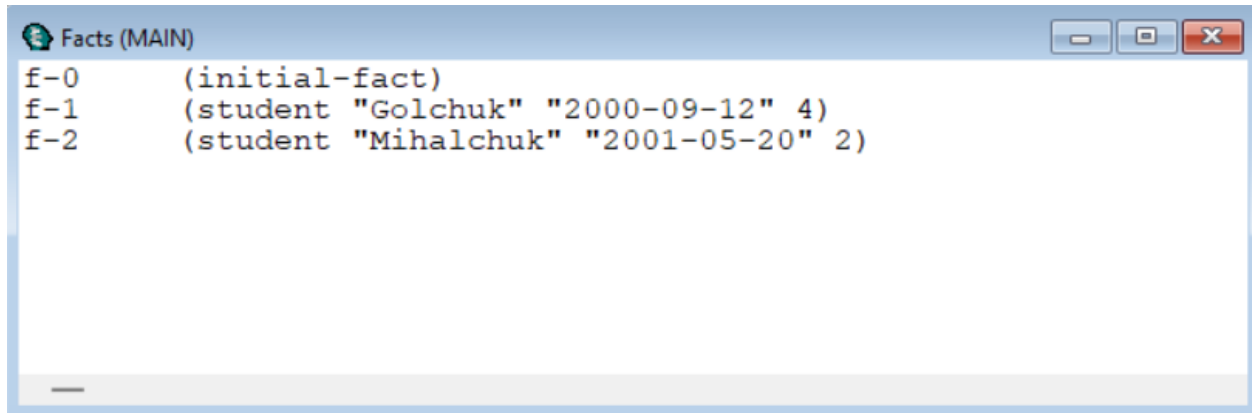


Рис. 9.

7. Видалив декілька фактів зі сформованого списку, використовуючи функцію retract (див Рис. 10).

```
CLIPS> (refact 2)
CLIPS> (refact 1)
```

Рис. 10.

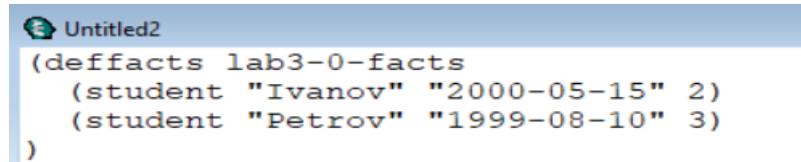
8. Використовуючи функцію fact-existp, переконався в існуванні фактів з вказаним індексом (див. Рис. 11).

```
CLIPS> (fact-existp 0)
TRUE
CLIPS>
```

Рис. 11.

9. Ввів у пам'ять масив впорядкованих фактів, використовуючи конструктор deffacts.

Для цього: – відкрив вікно вбудованого редактора в CLIPS; – написав код введення масиву фактів, використовуючи конструктор deffacts(див Рис. 12);



```
(deffacts lab3-0-facts
  (student "Ivanov" "2000-05-15" 2)
  (student "Petrov" "1999-08-10" 3)
)
```

Рис. 12.

– записав файл в пам'ять під іменем lab\_3(див. Рис. 13);

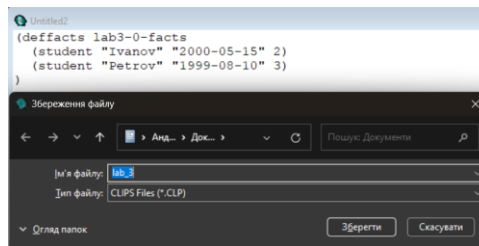
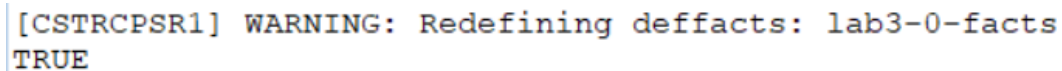


Рис. 13.

– завантажив файл в CLIPS за допомогою команди load (див. Рис. 14);



```
[CSTRCPSR1] WARNING: Redefining deffacts: lab3-0-facts
TRUE
```

Рис. 14.

– впевнився у правильності синтаксису коду введення масиву;

– переглянув список введених фактів(див Рис. 15)

```
CLIPS> (facts)
f-0      (initial-fact)
f-1      (student "Golchuk" "2000-09-12" 4)
f-2      (student "Mihalchuk" "2001-05-20" 2)
For a total of 3 facts.
```

Рис. 15.

10. Впевнився в виконанні команд, описаних у пп.4-8.

11. Створив список з 5 неупорядкованих фактів у вигляді списку студентів групи у вигляді:

(student (name <П\_І\_Б>) (група <група>) (date <дата народження>)),  
використовуючи конструктор deftemplate. Для цього: – я відкрив вікно  
вбудованого редактора CLIPS; – використовуючи конструктор deftemplate,  
створив текстовий файл з описом відповідного шаблону та записав файл в  
пам'ять під певним іменем (lab3-1-template)(див Рис. 16);  
CLIPS> (load "C:/Users/andri/OneDrive/Документи/lab3-1-template.CLP")  
Defining deftemplate: student  
TRUE

Рис. 16.

– створив текстовий файл з 5 фактів вказаного у шаблоні формату і записав у  
файл під новим іменем (lab3-1-facts1)(див Рис. 17);

CLIPS> (load "C:/Users/andri/OneDrive/Документи/lab3-1-facts1.CLP")

Рис. 17.

12. Використовуючи написані у вбудованому редакторі коди, введіть з нього в CLIPS шаблон та створені за його допомогою факти(див Рис8 19).

```
CLIPS> (load "C:/Users/andri/OneDrive/Документи/lab3-1-template.CLP")
Defining deftemplate: student
TRUE
CLIPS> (assert (student (name "Ivanov") (grupa "IT-21") (date "2001-05-13")))
==> f-1      (student (name "Ivanov") (grupa "IT-21") (date "2001-05-13"))
<Fact-1>
CLIPS> (assert (student (name "Petrov") (grupa "IT-21") (date "2002-04-12")))
==> f-2      (student (name "Petrov") (grupa "IT-21") (date "2002-04-12"))
<Fact-2>
CLIPS> (assert (student (name "Diganon") (grupa "IT-21") (date "1999-03-11")))
==> f-3      (student (name "Diganon") (grupa "IT-21") (date "1999-03-11"))
<Fact-3>
CLIPS> (assert (student (name "Miron") (grupa "IT-21") (date "2001-02-14")))
==> f-4      (student (name "Miron") (grupa "IT-21") (date "2001-02-14"))
<Fact-4>
CLIPS> (assert (student (name "Sifon") (grupa "IT-21") (date "2001-01-15")))
==> f-5      (student (name "Sifon") (grupa "IT-21") (date "2001-01-15"))
<Fact-5>
CLIPS> █
```

Рис 18.

13. Перегляньте список введених невпорядкованих фактів, використовуючи діалогове вікно, а також пункт 1 Fact Windows підменю Windows графічного інтерфейсу CLIPS(див Рис. 19).

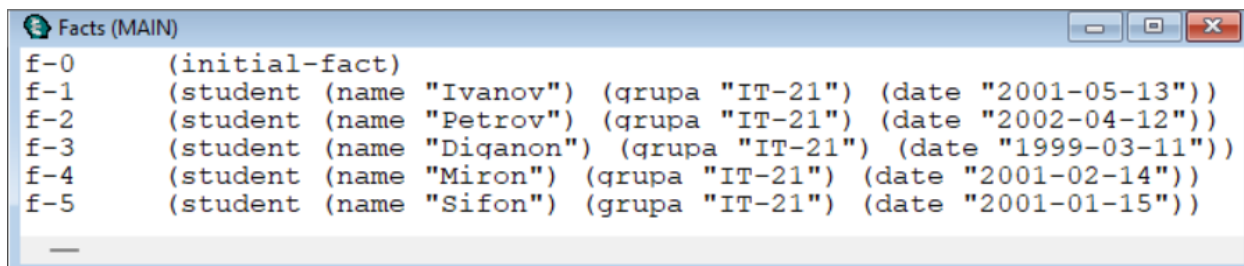


Рис. 19.

14. Модифікуйте декілька з введених невпорядкованих файлів; для модифікації фактів використайте команду modify(див. Рис. 20).

```
CLIPS> (modify 2 (name "Sidorov"))
<== f-2      (student (name "Petrov") (grupa "IT-21") (date "2002-04-12"))
==> f-6      (student (name "Sidorov") (grupa "IT-21") (date "2002-04-12"))
<Fact-6>
```

Рис. 20.

15. Продублюйте декілька записаних невпорядкованих фактів, змінивши поля слотів за допомогою команди `duplicate`(див Рис. 21).

```
CLIPS> (duplicate 3 (name "Petrov"))  
==> f-7      (student (name "Petrov") (grupa "IT-21") (date "1999-03-11"))  
<Fact-7>
```

Рис. 21.

16. Перевірів результат виконання команди з заборотою існування у базі знань ідентичних фактів і з дозволом існування таких фактів(див Рис.22).

```
CLIPS> (setfact-duplication TRUE)  
CLIPS> (duplicate 4)  
FALSE
```

Рис. 22.

17. Видалив декілька невпорядкованих фактів зі сформованого списку, використовуючи функцію `retract`(див Рис. 23).

```
CLIPS> (retract 6)  
<== f-6      (student (name "Sidorov") (grupa "IT-21") (date "2002-04-12"))  
CLIPS> (retract 5)  
<== f-5      (student (name "Sifon") (grupa "IT-21") (date "2001-01-15"))  
CLIPS> █
```

Рис. 23.

18. Переглянув список невпорядкованих фактів(див Рис. 24.).

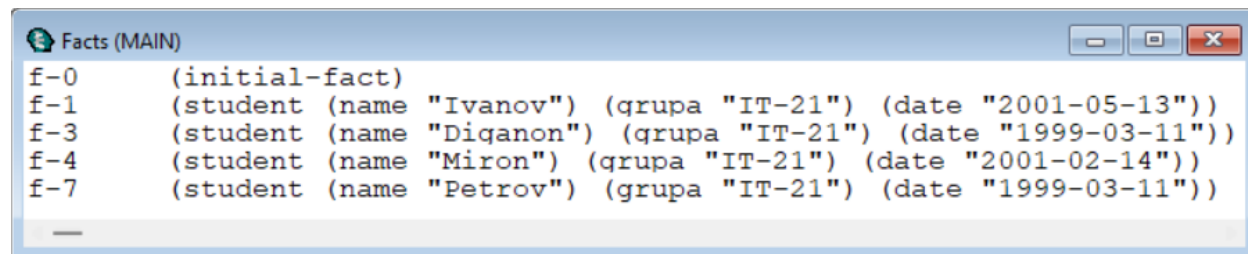


Рис. 24.

19. Записав внесені в CLIPS факти в файл під іменем lab3-facts2 (див. Рис. 25).

```
CLIPS> (save-facts "lab3-facts2.clp")  
TRUE
```

Рис. 25.

20. Повністю очистив пам'ять системи командою reset (див. Рис 26).

```
CLIPS> (reset)
```

Рис. 26.

21. Використовуючи записані раніше в файли конструкцію шаблону та створені відповідні неупорядковані факти, введіть їх в систему за допомогою команди load(див. Рис. 27).

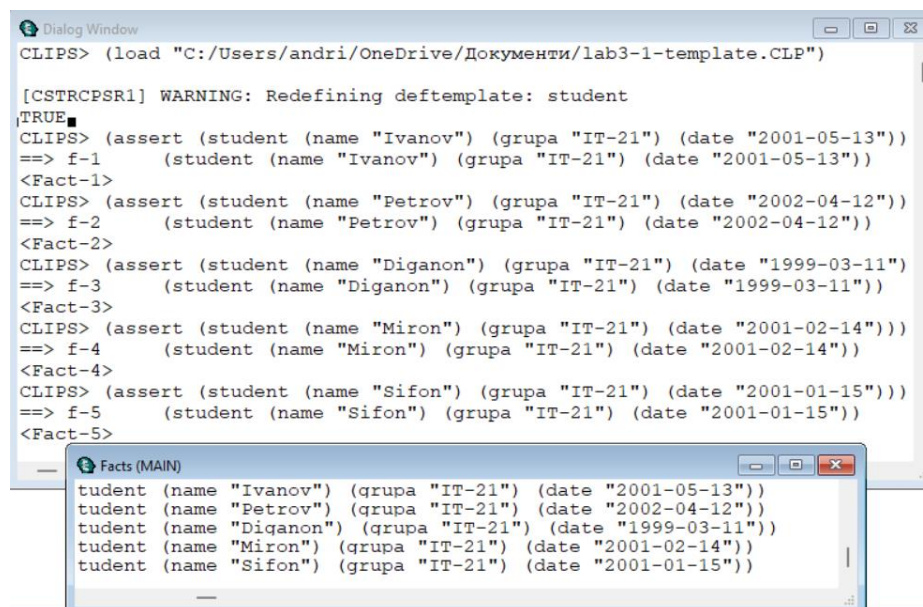


Рис. 27.



22. Пересвідчився у отриманні результату, аналогічного з отриманим використовуючи вбудований редактор(див. Рис. 28).

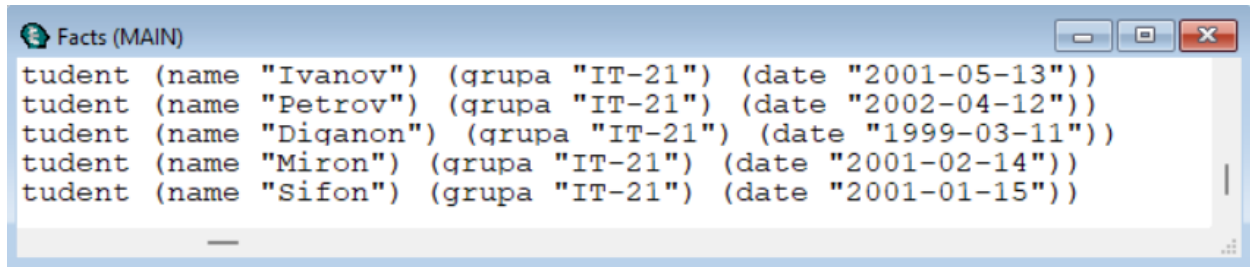


Рис. 28.

**Висновок:** У результаті виконаної роботи я зрозумів, наскільки важливою є правильна організація та маніпуляція фактами в системі CLIPS. Завдяки використаним командам для додавання, дублювання та видалення фактів, я побачив, як ці процеси можуть полегшити управління знаннями і прийняття рішень. Робота показала, що автоматизація таких завдань суттєво підвищує ефективність, особливо в рамках експертних систем.