МЕТОДИ І СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

3 курс, весна 2021

- Доц. Баклан І.В.
- Email: iaa@ukr.net
- Web: baklaniv.at.ua

Лекція 10

CLIPS. Факти.

Для функціонування будь-якої експертної системи критично важливим є наявність бази знань. Про це говорить навіть той факт, що останнім часом всі частіше термін "система, заснована на знаннях" (knowledge-base system) уживається як синонім терміна "експертна система". Як правило, у будь-якій експертній системі знання представляються фактами й правилами, заданими на деякій мові опису знань. CLIPS не є виключенням і надає можливості для придбання, зберігання й обробки фактів і правил. Дана лекція присвячена способам роботи з фактами в системі CLIPS. Робота із правилами буде висвітлюватися в наступних лекціях.

Факти в CLIPS

Факти — одна з основних форм подання даних в CLIPS (існує також можливість подання даних у вигляді об'єктів і глобальних змінних, але про це мова йтиме пізніше). Кожен факт являє собою певний набір даних, що зберігає в поточному списку фактів - робочої пам'яті системи. Список фактів являє собою універсальне сховище фактів й є частиною бази знань. Об'єм списку фактів обмежений тільки пам'яттю вашого комп'ютера. Список фактів зберігається в оперативній пам'яті комп'ютера, але CLIPS надає можливість зберігати поточний список у файл і завантажувати список з раніше збереженого файлу.

У системі CLIPS фактом є список неподільних (або атомарних) значень примітивних типів даних. CLIPS підтримує два типи фактів - ***упорядковані факти*** (ordered facts) і ***неупорядковані факти*** або *шаблони* (non-ordered facts або template facts). Посилатися на дані, що втримуються у факті, можна або використовуючи строго задану позицію значення в списку даних для впорядкованих фактів, або вказуючи ім'я значення для шаблонів.

Факти можна додавати, видаляти, змінювати й дублювати, уводячи відповідні команди із клавіатури, або із програми. Всі відповідні команди будуть описані в даній лекції.

Після додавання факту в список фактів йому привласнюється цілий унікальний ідентифікатор, називаний ***індексом факту*** (fact-index). Індекс першого факту дорівнює нулю, надалі індекс збільшується на одиницю при додаванні кожного нового факту. CLIPS надає команди, що очищають поточний список фактів або всю базу знань, ці команди привласнюють поточному значенню індексу 0.

Деякі команди, наприклад зміни, видалення або дублювання фактів, вимагають вказівки певного факту. Факт можна задати або індексом факту, або його адресою. Адреса факту являє собою змінн-покажчик, що зберігає індекс факту. Процес створення адрес фактів буде описаний нижче.

Упорядковані факти складаються з поля, що обов'язково є даним типу **`symbol`** і наступної за ним, можливо порожньої, послідовності полів, розділених пробілами. Обмеженням факту служать круглі дужки.

Визначення 10.1. Упорядкований факт

`(дане_типу_symbol  [поле]*)`

Перше поле факту визначає так називане ***відношення***, або *зв'язок* факту (relation). Термін "зв'язок" означає, що даний факт належить деякому певному конструктором або неявно оголошеному шаблону. Докладніше мова про це піде нижче.

Приведемо кілька прикладів фактів:

Приклад 10.1 Упорядковані факти

```
(duck is bird)
(schoolboys is Bob Mike)
(Nuke did report)
(altitude is 1000 feet)
```

Кількість полів у факті не обмежено. Поля у факті можуть зберігати дані будь-якого примітивного типу CLIPS, за винятком першого поля, що обов'язково повинне бути типу **`symbol`**. Наступні слова зарезервовані й не можуть бути використані як перше поле: **`test`**, **`and`**, **`or`**, **`not`**, **`declare`**, **`logical`**, **`object`**, **`exist`** й **`forall`**. Ці слова можуть використатися як імена слотів шаблонів, хоча це не рекомендується.

Тому що впорядкований факт для подання інформації використає строго задані позиції даних, то для доступу до неї користувач повинен знати не тільки які дані збережені у факті, але і яке поле містить ці дані. Неупорядковані факти (або шаблони) надають користувачеві можливість задавати абстрактну структуру факту шляхом призначення імені кожному полю. Для створення шаблонів, які згодом будуть застосовуватися для доступу до полів факту по імені, використається конструктор **`deftemplate`**. Конструктор **`deftemplate`** аналогічний визначенням записів або структур у таких мовах програмування, як Pascal або C.

Конструктор **deftemplate** задає ім'я шаблона й визначає послідовність із нуля або більше полів неупорядкованого факту, називаних також ***слотами***. Слот складається з імені, заданого значенням типу **symbol**, і наступної за ним, можливо порожнього, списку полів. Як і факт, слот по обидва боки обмежується круглими дужками. На відміну від упорядкованих фактів слот неупорядкованого факту може жорстко визначати тип своїх значень. Крім того, слоту можуть бути задані значення за замовчуванням.

Зауваження

Слоти не можуть бути використані в упорядкованих фактах, а в неупорядкованих фактах, у свою чергу, не можна посилатися на дані, використовуючи порядок слотів.

CLIPS розрізняє неупорядковані факти від упорядкованих по першому полю факту. Перше поле фактів будь-якого типу є значенням типу **`symbol`**. Якщо це значення відповідає імені деякого шаблона, то факт - упорядкований. Визначення неупорядкованого факту, як й упорядкованого, обмежується круглими дужками.

Нижче наведено кілька прикладів неупорядкованих фактів.

Приклад 10.2. Неупорядковані факти

```
(client   (name  "Joe Brown")   (id X9345A))
(point-mass   (x-velocity 100)    (y-velocity
-200))
(class    (teacher   "Martha Jones")      (#-
students  30)    (Room "37A")
(grocery-list   (#-of-items 3)    (items bread
milk eggs))
```

Зауваження

Порядок слотів *у* неупорядкованому факті не важливий. Наприклад, всі наведені нижче факти вважаються ідентичними:

```
(class (teacher "Martha Jones")(#-students 30)
(Room "37A"))
(class (#-students 30)(teacher "Martha Jones")
(Room "37A"))
(class (Room "37A")(#-students 30)(teacher
"Martha Jones"))
```

На відміну від фактів, наведених вище, упорядковані факти з наступного приклада не є ідентичними:

```
(class "Martha Jones" 30 "37A")
(class 30 "Martha Jones" "37A")
(class "37A" 30 "Martha Jones")
```

З неупорядкованими фактами можна виконувати ті ж операції, що й з упорядкованими.

Далі розглянемо конструктори, операції й функції, які надає CLIPS для роботи з фактами.

Робота з фактами

CLIPS надає досить багатий набір можливостей для роботи з фактами за допомогою відповідних конструкторів, операцій і функцій. Ці можливості включають створення шаблонів за допомогою конструктора `deftemplate`, створення, зміна, видалення, пошук фактів, перегляд, збереження й завантаження списку фактів, визначення списку визначених фактів за допомогою конструктора deffacts і багато чого іншого.

Конструктор *deftemplate*

Для створення неупорядкованих фактів в CLIPS передбачений спеціальний конструктор **`deftemplate`**. Його використання приводить до появи в поточній базі знань системи інформації про шаблон факту, за допомогою якого в систему надалі можна буде додавати факти, що відповідають даному шаблону. Таким чином, конструктор **`deftemplate`** аналогічний операторам **`record`** й **`struct`** таких процедурних мов програмування як Pascal або C.

Приведемо простий приклад використання конструктора **deftemplate**:

Приклад 10.3. Застосування конструктора deftemplate

```
(deftemplate MyObject
     (slot name)
     (slot location)
     (slot weight)
     (multislot contents))
```

Як і всі конструктори CLIPS, конструктор **`deftemplate`** не повертає ніякого значення. При уведенні даної команди в CLIPS ви повинні побачити результат, наведений на мал. 10.1.

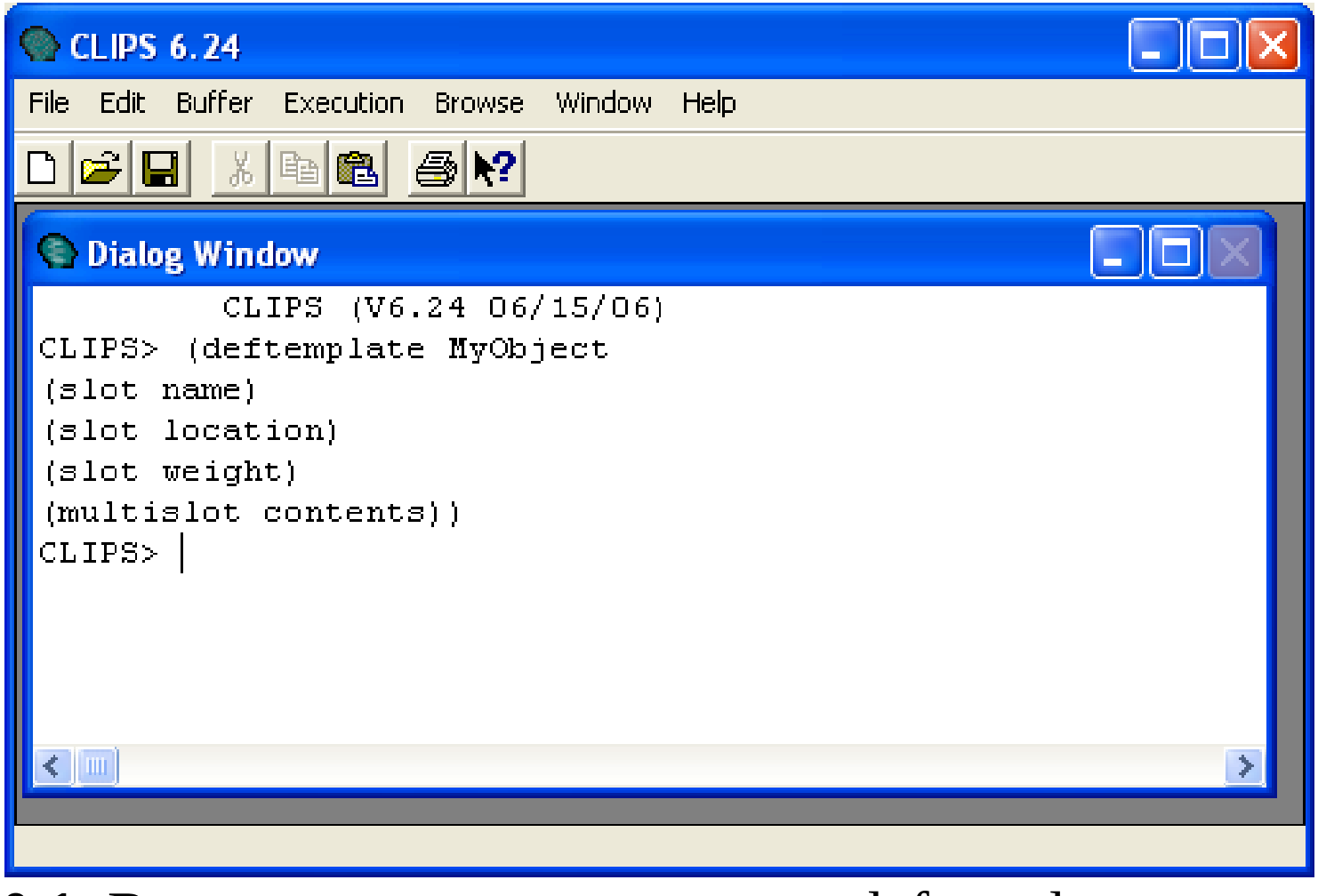

Рис. 10.1. Використання конструктора deftemplate

Подібна реакція середовища говорить про вдале додавання визначення шаблона в систему. Для перегляду всіх певних у поточній базі знань шаблонів можна скористатися командою **`get-deftemplate-list`**, мова про яку піде нижче, або спеціальним інструментом **`Deftemplate Manager`** (Менеджер шаблонів), доступним в Windows-версії середовища CLIPS. Для запуску менеджера шаблонів скористайтеся меню **`Browse`** і виберіть пункт **`Deftemplate Manager`** (мал. 10.2).

Менеджер шаблонів дозволяє в окремому вікні переглядати список всіх шаблонів, доступних у поточній базі знань, видаляти обраний шаблон і відображати всієї його властивості (наприклад, такі як імена й типи слотів). Зовнішній вигляд менеджера шаблонів представлений на мал. 10.3.

Після виконаної нами операції в поточній базі знань перебуває два шаблони, про що повідомляється в заголовку вікна менеджера (**Deftemplate Manager — 2 Items).** Перший шаблон є визначеним шаблоном **initial-fact**. Він не має слотів і завжди додається при запуску середовища. Його не можна видалити за допомогою менеджера, або переглянути його визначення. Призначення й приклади використання факту **initial-fact** будуть розглянуті нижче. Другим шаблоном є тільки що доданий шаблон **MyObject**. Менеджер шаблонів дозволяє вивести в головне вікно середовища його визначення за допомогою кнопки **Pprint** або видалити його із середовища за допомогою кнопки **Remove.** На мал. 10.4 наведений результат послідовних операцій висновку інформації про визначення шаблона й видаленні його з поточної бази знань.

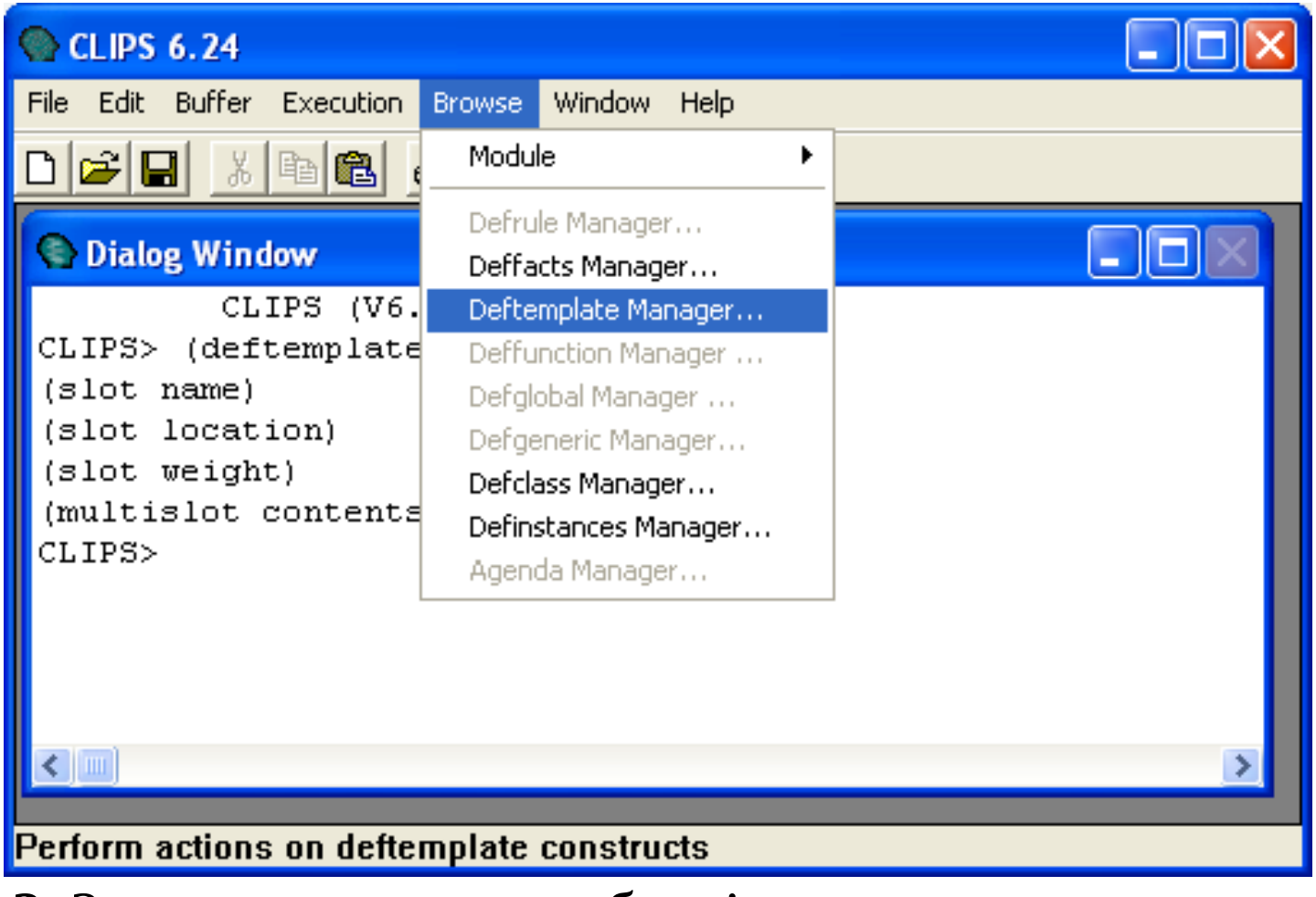

Рис. 10.2. Запуск менеджера шаблонів

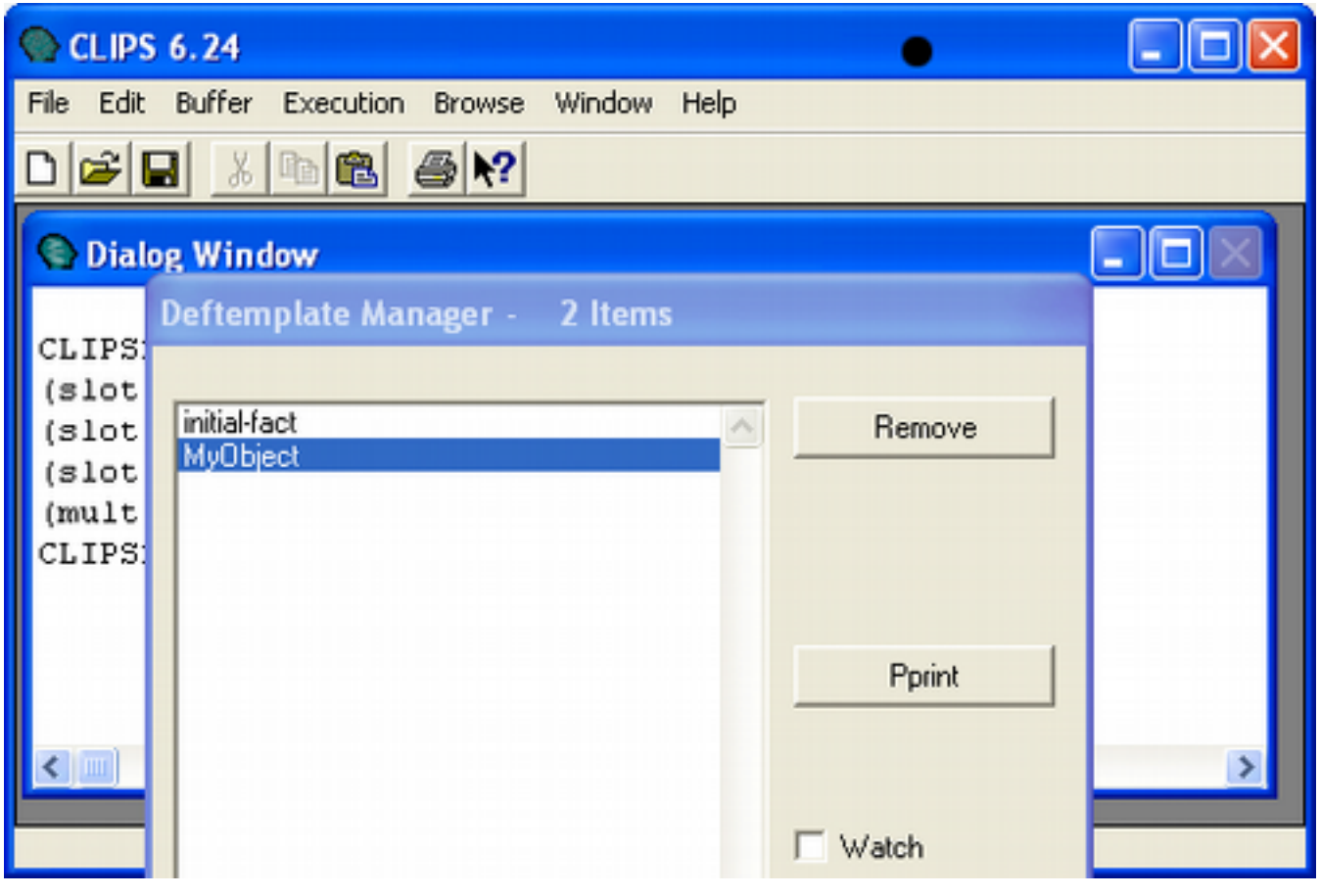

Рис. 10. 3. Вікно менеджера шаблонів

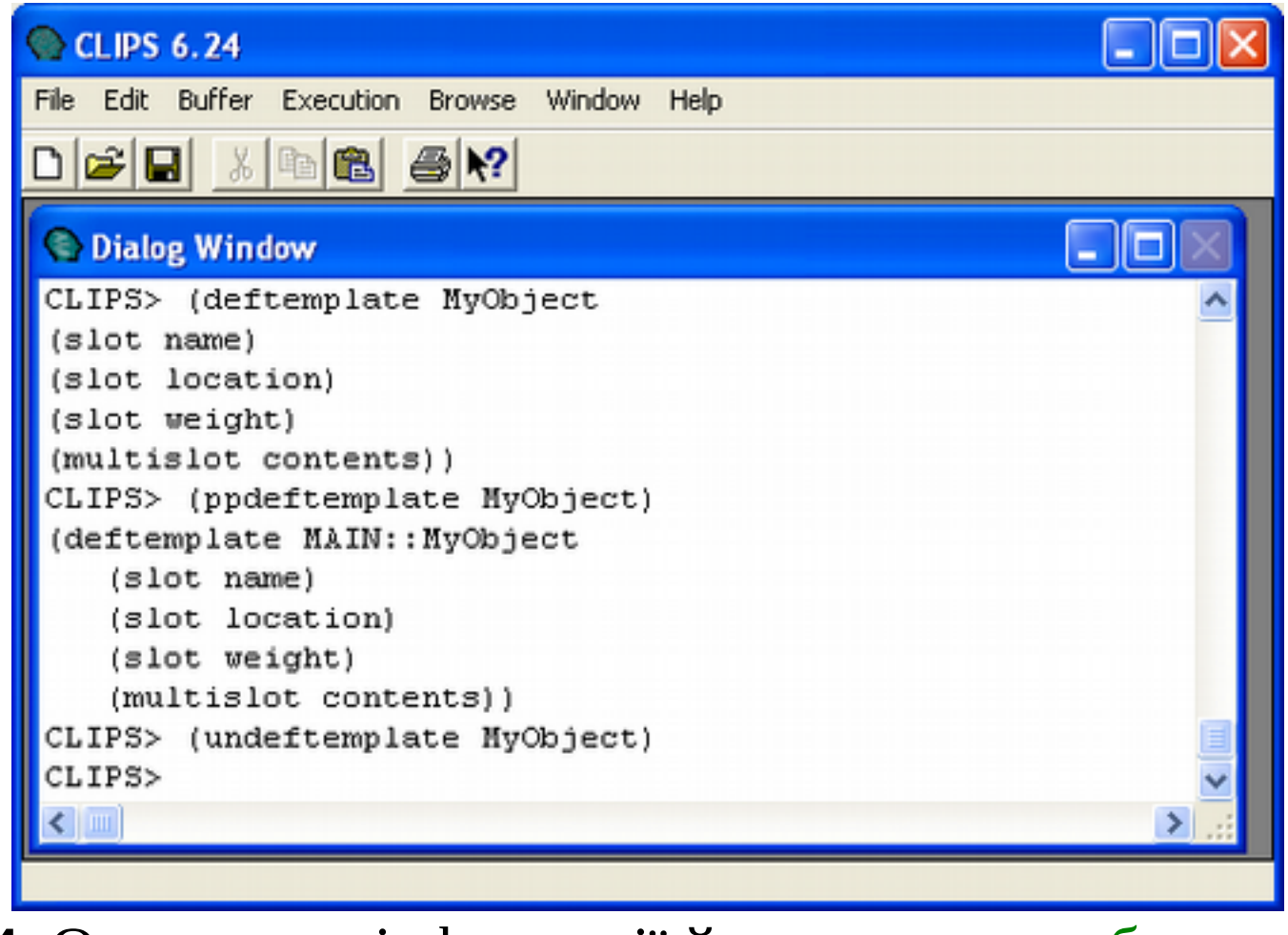

Рис. 10.4. Одержання інформації й видалення шаблона

Прапорець **Watch** дозволяє включати/виключати режим відображення повідомлень про використання шаблонів для кожного присутнього в системі шаблона в головному вікні середовища CLIPS. Якщо цей режим включений, користувач буде одержувати повідомлення при додаванні й видаленні неупорядкованих фактів, що використають даний шаблон.

У випадку, якщо при додаванні нового шаблона за допомогою конструктора **deftemplate** відбулася помилка, користувач одержить відповідне попередження. .

Перевизначення вже існуючого шаблона приводить до виключення попереднього визначення. Шаблон не може бути перевизначений доти, поки він використається (наприклад, фактом або правилом). Шаблон може мати будь-яка кількість простих або складених слотів. CLIPS відрізняє прості й складені слоти в шаблоні. Наприклад, буде помилкою зберігати значення складеного слоту в простий слот.

Розглянемо повний синтаксис конструктора **deftemplate**:

Визначення 10.2. Синтаксис конструктора deftemplate

```
(deftemplate    <імені-шаблона>[<необов'язков-коментарі>] [<определение-слота>*])

<определение-слота>                                    ::=
<определение-простого-слота>|<определение-составного-слота>

<определение-простого-слота>          ::=  (slot <ім'я-поля> <атрибута-шаблона>)

<определение-составного-слота>                         ::=
```

(multislot <ім'я-поля> <атрибута-шаблона>)

<атрибута-шаблона> ::=
<атрибуту-значенню-по-умовчанню>|<атрибута-обмеження>

<атрибуту-значенню-по-умовчанню> ::= (default ?DERIVE I ?NONE |<Вираз>)| (default-dynamic <Вираз>)

Помітимо ще раз, що імена шаблонів і слотів повинні бути значеннями типу **symbol**, крім того, на імена шаблонів поширюється заборона на використання деяких слів, зарезервованих системою, перерахованих вище.

Коментарі є необов'язковими й, як правило, описують призначення шаблона. Коментарі необхідно містити в лапки. Крім даного типу коментарів у конструкторі **deftempiate** також застосовні звичайні коментарі CLIPS, що починаються із символу ; . Відмінність цих коментарів полягає в тім, що коментарі, що починаються із символу ; , повністю ігноруються системою CLIPS, а коментарі, що випливають після імені шаблона й ув'язнені в лапки, зберігаються в базі знань системи. Ці коментарі доступні при перегладі визначення шаблона.

Визначимо в середовищі CLIPS наступний шаблон:

Приклад 10.4. Застосування конструктора deftemplate

```
(deftemplate MyObject "Template for storage
name and location" ; Slots for storage
name and location
 (slot name) ;slot for name of object
 (slot location) ; slot for location of object
```

Після вдалого додавання шаблона в систему, за допомогою менеджера шаблонів, виведіть у головне вікно CLIPS інформацію про визначення шаблона **`MyObject`**. Якщо перераховані дії були виконані без помилок, то на екрані повинні з'явитися повідомлення, ідентичні показаним на мал. 10.5.

Зверніть увагу, що коментарі ***`"Template for storage name and location"`*** збережені в пам'яті системи й відображаються разом з визначенням шаблона. На жаль, що ця версія CLIPS не сприймає символи кирилиці навіть як коментарі, тому всі коментарі прийде давати англійською мовою.

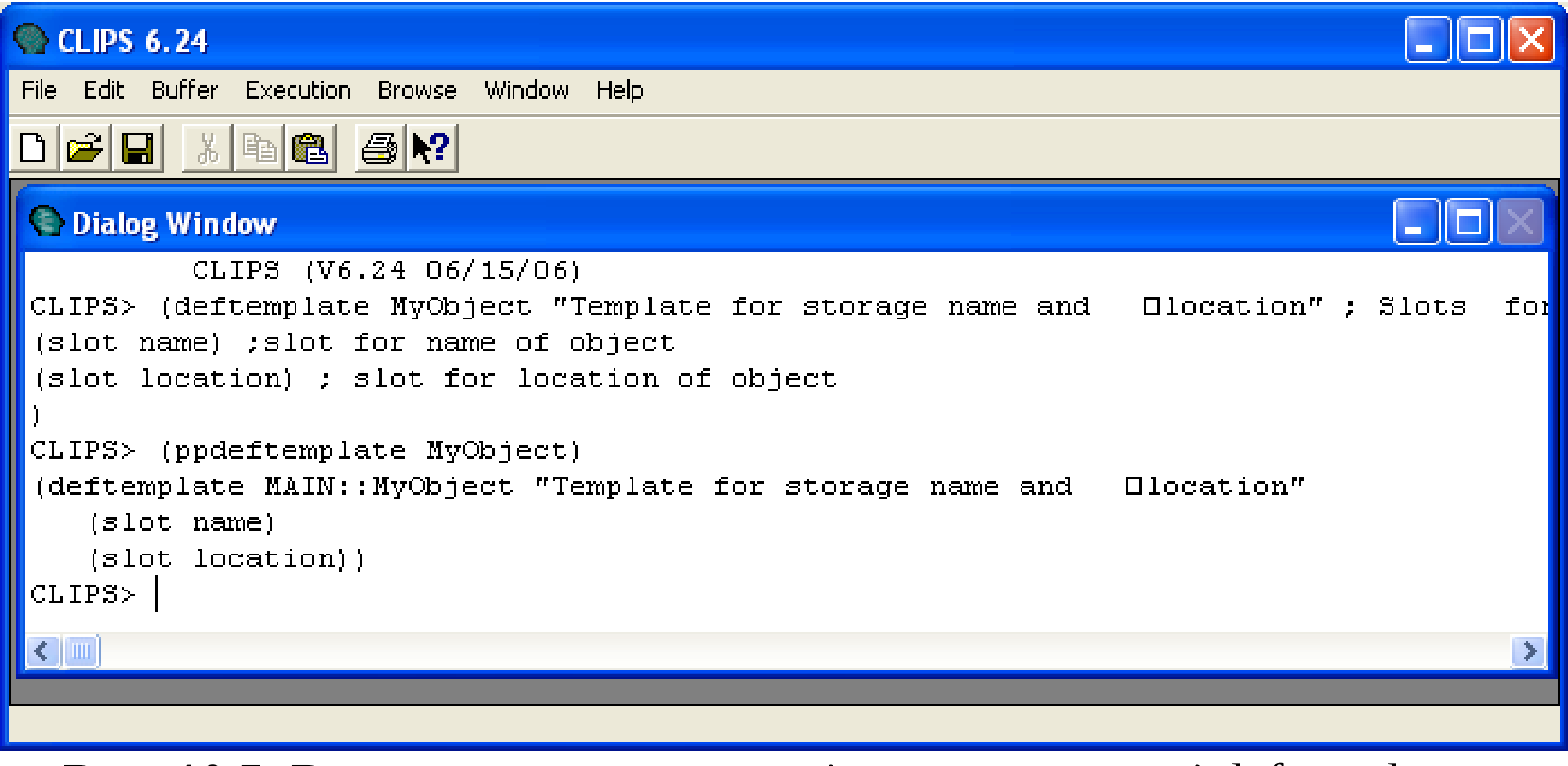

Рис. 10.5. Використання коментарів у конструкторі deftemplate

Крім ключового слова **slot**, що визначає простий слот, припустиме також застосування ключового слова **multisiot**, для визначення складеного слоту. Простий слот, або слот, призначений для зберігання одиниці інформації одного із примітивних типів даних CLIPS. Складений слот здатний зберігати список подібних одиниць інформації необмеженого об'єму. Для доступу до конкретних даних, що зберігається в складеному слоті, використаються спеціальні групові символи й функції, приклади й правила використання яких будуть наведені нижче.

При створенні шаблона за допомогою конструктора **deftemplate** кожному полю можна призначати певні атрибути, що задають значення за замовчуванням або обмеження на значення слоту. Розглянемо ці атрибути детальніше.

<Атрибут-значення-за-замовченням> визначає значення, що буде використано у випадку, якщо при створенні факту не задане конкретне значення слоту. В CLIPS існує два способи визначення значення за замовчуванням, тому в конструкторі **deftemplate** передбачено два різних атрибути, що задає значення за замовчуванням: **default** й **default-dynamic**.

Атрибут **default** визначає статичне значення за замовчуванням. З його допомогою задається вираження, що обчислюється один раз при конструюванні шаблона. Результат обчислень зберігається разом із шаблоном. Цей результат привласнюється відповідному слоту в момент оголошення нового факту. У випадку якщо як значення за замовчуванням використається ключове слово **?DERIVE**, те це значення буде витягатися з обмежень, заданих для даного слоту.

За замовчуванням для всіх слотів установлений атрибут **DEFAULT ?DERIVE**.

В випадку якщо в місце вираз для значення за замовчуванням використається ключове слово **?NONE**, те значення поля обов'язково повинне бути явно задане в момент виконання операції додавання факту. Додавання факту без визначення значень полів з атрибутом **DEFAULT ?NONE** викличе помилку.

Атрибут **DEFAULT-DYNAMIC** призначений для установки динамічного значення за замовчуванням. Цей атрибут визначає вираз, що обчислюється щораз при додаванні факту по даному шаблоні. Результат обчислень привласнюється відповідному слоту.

Простий слот може мати тільки одне значення за замовчуванням. У складеного слоту може бути визначена будь-яка кількість значень за замовчуванням (кількість значень за замовчуванням повинне відповідати кількості даних, що зберігають у складеному слоті).

Нижче наведений приклад використання атрибута, що встановлює значення за замовчуванням:

Приклад 10.5. Використання атрибутів значення за замовчуванням

```
(deftemplate foo
     (slot w   (default  ?NONE))
     (slot x   (default  ?DERIVE))
     (slot y   (default   (gensym*)))
     (slot z   (default-dynamic   (gensym*))))
```

У конструкторі **deftemplate** підтримується перевірка статичних і динамічних обмежень.

Статична перевірка виконується під час використання визначення шаблона деякою командою або конструктором. Наприклад, для запису значень у слоти шаблона. Інакше кажучи, статична перевірка виконується до запуску програми. При невідповідності використовуваних значень із установленими обмеженнями користувачеві виводиться відповідне попередження про помилку.

Посилання на індекс факту в командах на зміну значення факту або його дублювання не зв'язує факт із відповідним шаблоном явно. Це робить статичну перевірку неоднозначної. Тому в командах, що використають індекс факту, статична перевірка не виконується.

Статична перевірка обмежень включена за замовчуванням. Цю установку середовища CLIPS можна змінити за допомогою функції **`set-static-constraint-checking`**.

Крім статичної, CLIPS також підтримує динамічну перевірку обмежень. Якщо режим динамічної перевірки обмежень включенийі, го всі нові факти, створені з використанням деякого шаблона й певні значення, що мають, перевіряються в момент їхнього додавання в список фактів.

У випадку якщо порушення заданих обмежень відбудеться в момент виконання динамічної перевірки в процесі виконання програми, то виконання програми припиняється й користувачеві буде видане відповідне повідомлення.

За замовчуванням в CLIPS відключений режим динамічної перевірки обмежень. Це середовище установки можна змінити за допомогою функції **`set-dynamic-constraint-checking`**.

Крім описаних вище функцій для зміни станів режимів статичної й динамічної перевірки обмежень, користувачам Windows-версії середовища CLIPS доступний візуальний спосіб настроювання цих установок. Для цього необхідно відкрити діалогове вікно **Execution Options**, вибравши пункт **Options** з меню **Execution**. Зовнішній вигляд цього діалогового вікна наведений на мал. 10.6.

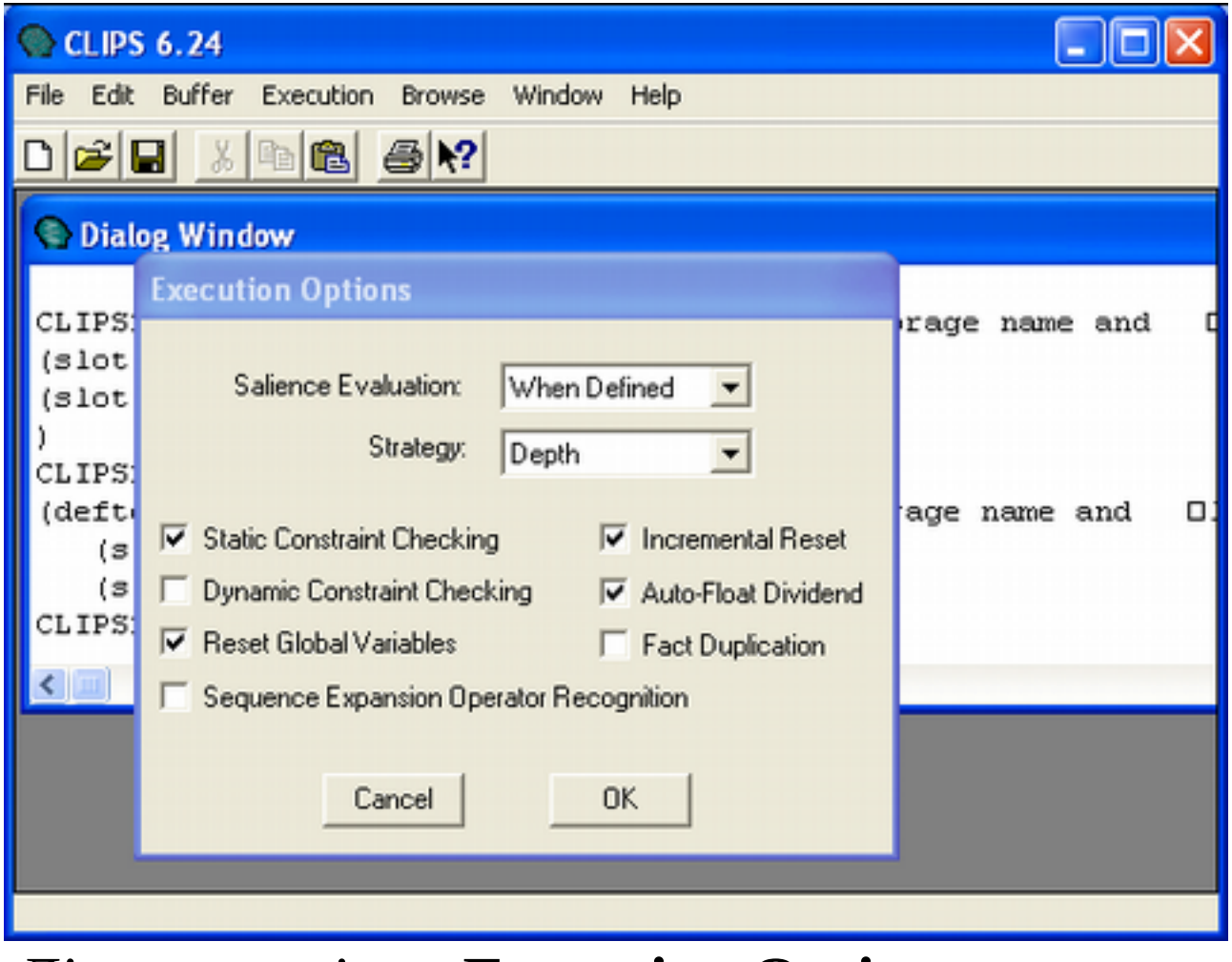

Рис. 10.6. Діалогове вікно **Execution Options**

Для включення або вимикання необхідних режимів перевірки обмежень атрибутів виставите у відповідне положення прапорці **Static Constraint Checking** й **Dynamic Constraint Checking** і натисніть кнопку **OK**.

Нижче наведений приклад використання атрибутів обмеження типу:

Приклад 10.6. Використання атрибутів обмеження

```
(deftemplate object
     (slot name
          (type SYMBOL)
          (default  ?DERIVE))
     (slot location
          (type SYMBOL)
          (default ?DERIVE)))
```

Для повноти картини варто також згадати про неявно створюваних шаблонах. При використанні факту або посилання на впорядкований факт (наприклад, у правилі) CLIPS неявно створює відповідний шаблон з одним складеним слотом. Ім'я неявно створеного складеного слоту не відображається при перегляді фактів. Неявно створеним шаблоном можна маніпулювати й порівнювати його з будь-яким тотожним, певним користувачем шаблоном, незважаючи на те, що він не має відображуваної форми.

Конструктор *deffacts*

Крім конструктора **`deftemplates`**, CLIPS надає конструктор **`deffacts`**, також призначений для роботи з фактами. Даний конструктор дозволяє визначати список фактів, які будуть автоматично додаватися щораз після виконання команди **`reset`**, що очищає поточний список фактів. Факти, додані за допомогою конструктора **`deffacts`**, можуть використатися й віддалятися так само, як і будь-які інші факти, додані в базу знань користувачем або програмою, за допомогою команди **`assert`**.

Визначення 10.3. Синтаксис конструктора deffacts

```
(deffacts   <імена-списків-фактів>
            [<необов'язкові-коментарі>]
            [<факт>*])
```

Додавання конструктора **deffacts** з ім'ям уже існуючого конструктора приведе до видалення попереднього конструктора, навіть якщо новий конструктор містить помилки. У середовищі CLIPS можлива наявність декількох конструкцій **deffacts** одночасно й будь-яке число фактів у них (як упорядкованих, так і неупорядкованих). Факти всіх створених користувачем конструкторів **deffacts** будуть додані при ініціалізації системи.

Всі зауваження із приводу використання коментарів у конструкторі **deftemplate** застосовні й до конструктора **deffacts**.
У поля факту можуть бути включені динамічні вираз, значення яких будуть обчислюватися при додаванні цих фактів у поточну базу знань CLIPS.

Приклад 10.7. Використання конструктора deffacts

```
(deffacts   startup "Refrigerator Status"
     (refrigerator light on)
     (refrigerator door open)
     (refrigerator temp (+ 5 10 15))
```

Зверніть увагу, що третій факт містить вираз, у даному прикладі суму трьох констант, але як вираз, ініціюючого значення факту, можуть використатися й більше складні вираз, наприклад, виклики функцій CLIPS або функцій, певних користувачем.

Перевірити роботу конструктора **deffacts** можна скориставшись діалогом **Watch Options**. Для цього виберіть пункт **Watch** меню **Execution** або використайте комбінацію клавіш **<Ctrl>+<W>**. У діалоговому вікні **Watch Options** включите режим перегляду зміни списку фактів, поставивши галочку в поле **Facts**, як показано на мал. 10.7.

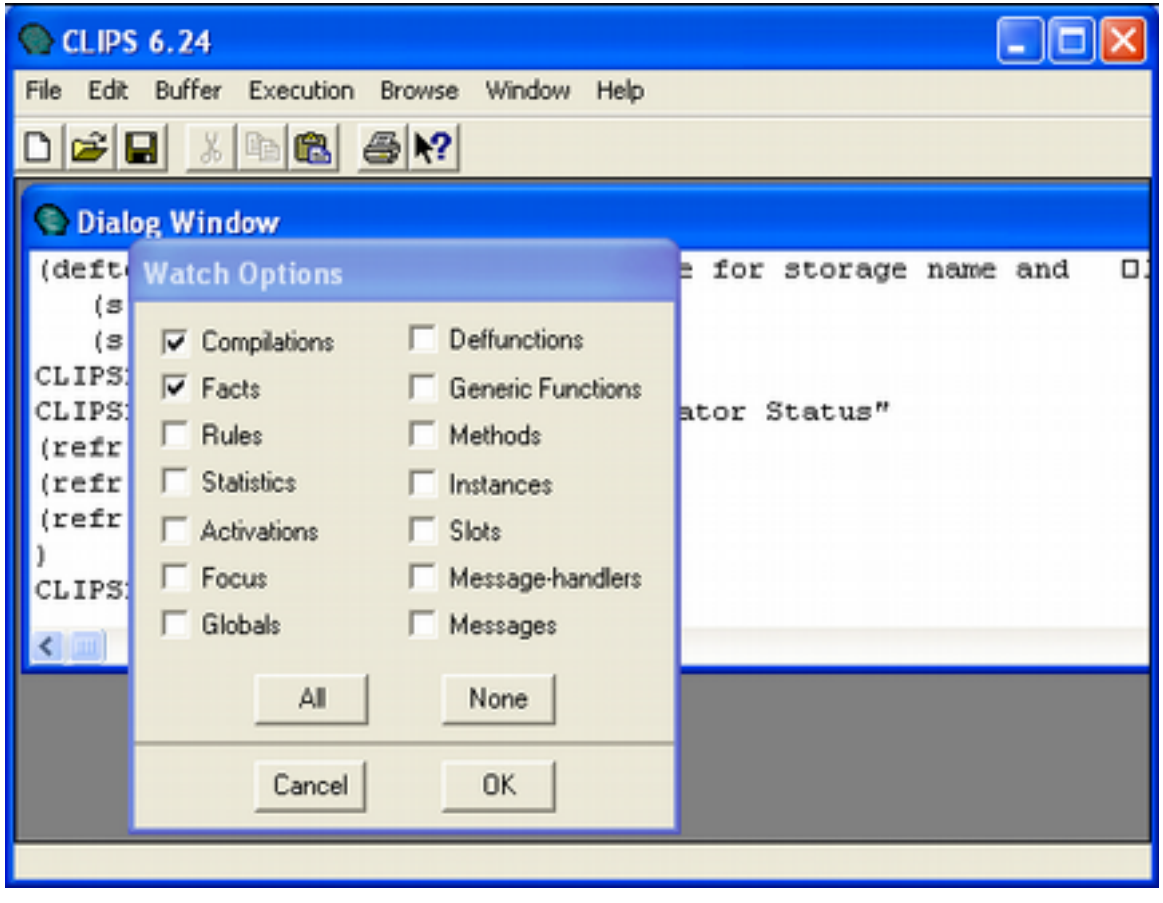

Рис. 10.7. Діалогове вікно **Watch Options**

Після цього натисніть кнопку OK й уведіть в CLIPS наведений вище конструктор **deffacts**. Потім у меню **Execution** виберіть пункт **Reset** (комбінація клавіш **<Ctrl>+<E>**). Якщо приклад був набраний правильно, то на екрані повинні з'явитися повідомлення, аналогічні наведеним на мал. 10.8.

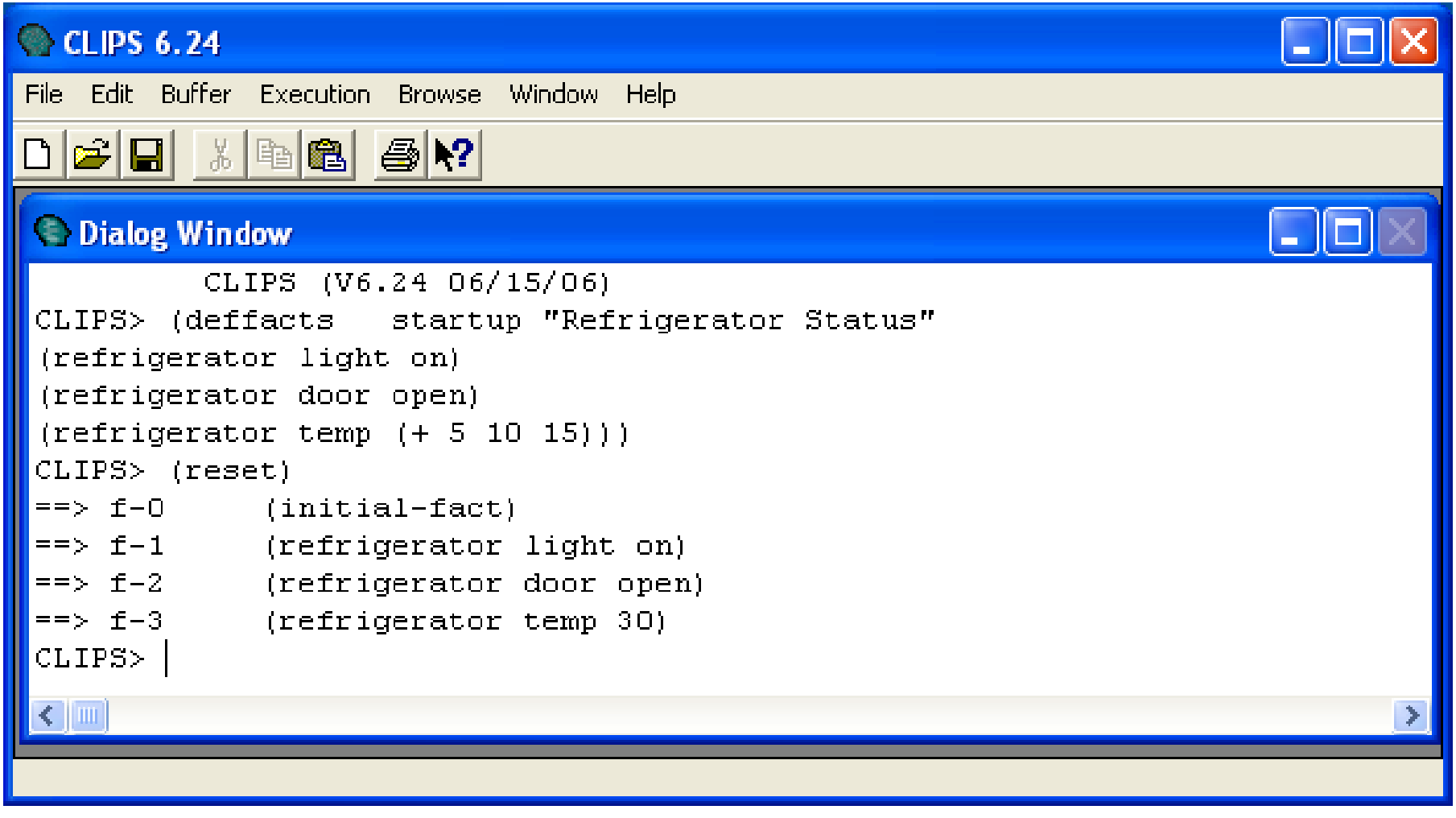

Рис. 10.8. Перегляд процесу додавання файлів

Так само, як і для конструкторів **deftemplate**, CLIPS надає візуальний інструмент для маніпуляції з певними в цей момент у системі конструкторами **deffacts** -- **Deffacts Manager** (Менеджер визначених фактів). Для запуску **Deffacts Manager** у меню **Browse** виберіть пункт **Deffacts Manager**. Зовнішній вигляд менеджера наведений на мал. 10.9.

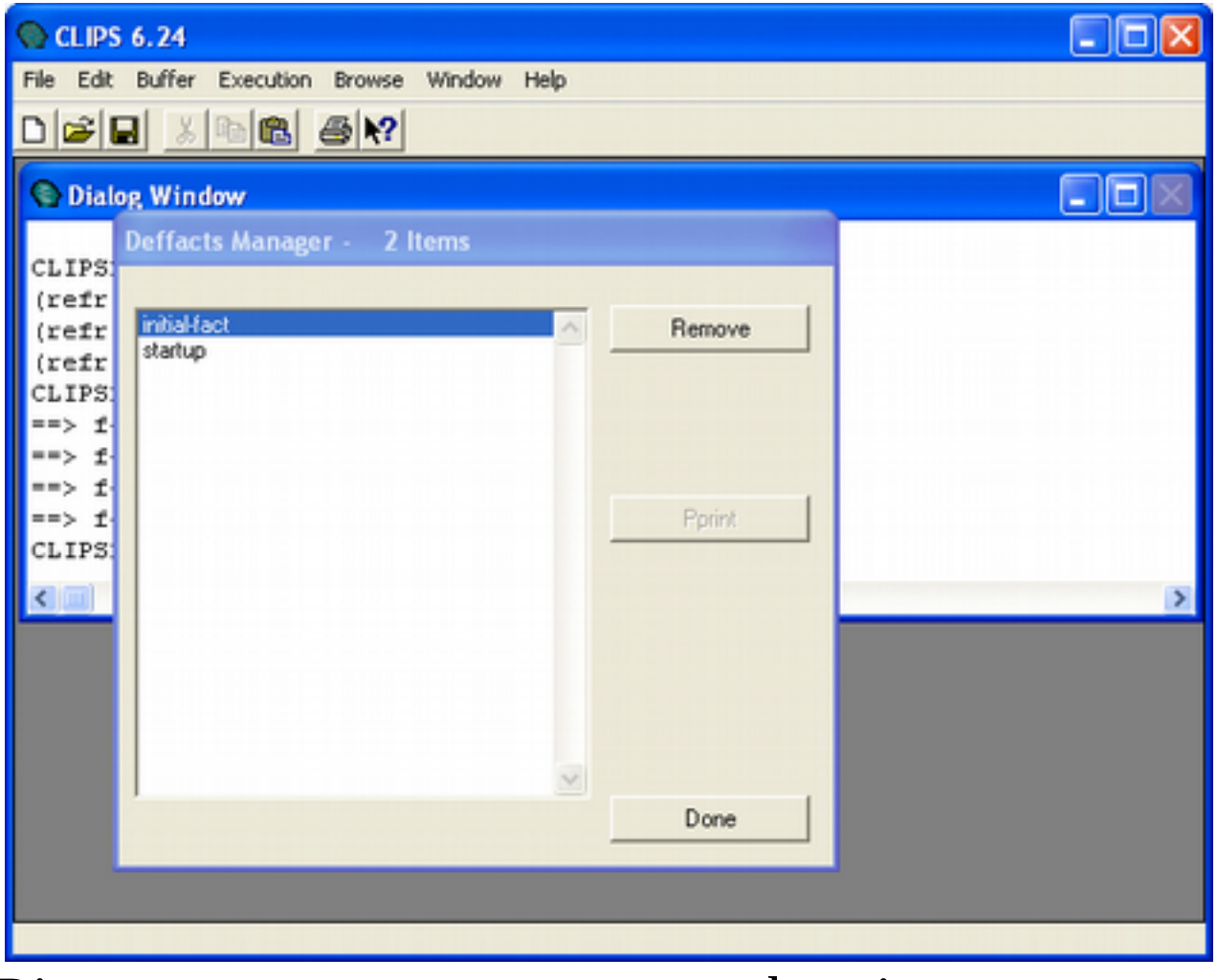

Рис. 10.9. Вікно менеджера визначених фактів

Менеджер відображає всі уведені на сучасний момент у систему конструктори **deffacts**. У нашому випадку це **initial-fact**, мова про яке піде нижче, і тільки що доданий нами **startup**. Менеджер дозволяє виводити в основне вікно CLIPS інформацію про визначення існуючих у цей момент у системі конструкторів **deffacts** за допомогою кнопки **Pprint** (крім **deffacts initial-fact**) і видаляти будь-який існуючий конструктор. Приклад висновку інформації про визначення конструктора **deffacts startup** наведений на мал. 10.10. Зверніть увагу, що коментарі, уведені після імені конструктора, зберігаються й виводяться на екран так само, як у конструкторі **deftemplate**.

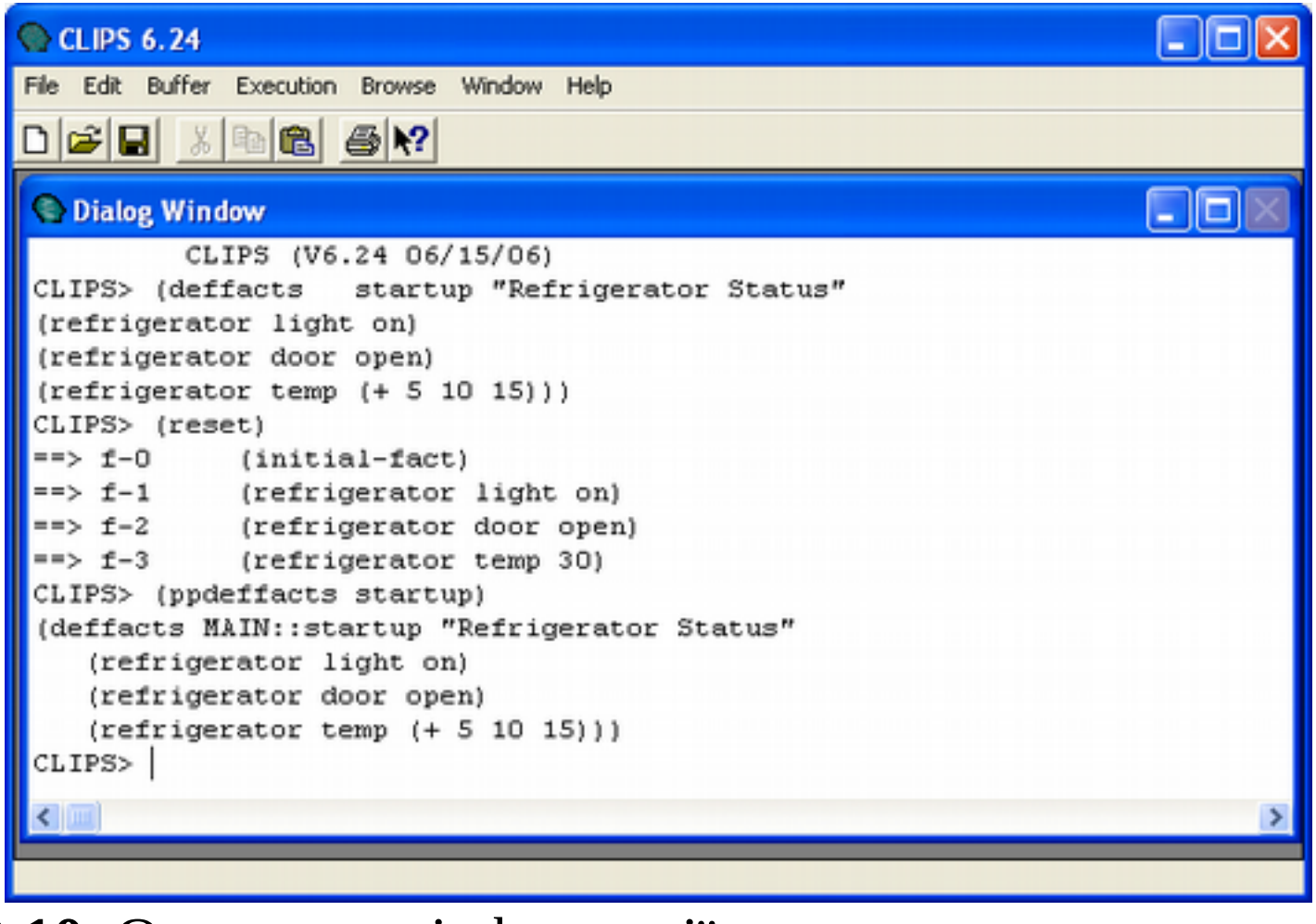

Рис.10.10. Одержання інформації про певного конструктора

Під час запуску й після виконання команди **`clear`** CLIPS автоматично конструює наступні визначені шаблони й факти:

Визначення 10.4. Визначені шаблони й факти

```
(deftemplate    initial-fact)
 (deffactsinitial-fact
    (initial-fact))
```

Визначений факт **`initial-fact`** шаблона **`initial-fact`** надає зручний спосіб для запуску програм мовою CLIPS — правила, що не мають умовних елементів, автоматично перетворяться в правила з умовою, що перевіряє наявність факту **`initial-fact`**. Факт **`initial-fact`** можна обробляти так само, як і всі інші факти CLIPS, додані користувачем або програмою за допомогою команди **`assert`**. Приклад використання факту **`initial-fact`** буде наведений у наступній лекції, відразу після першого знайомства із правилами CLIPS.

Функція *assert*

Функція assert - одна з найбільше часто застосовних команд у системі CLIPS. Без використання цієї команди не можна написати навіть найпростішу експертну систему й запустити її на виконання в середовищі CLIPS. Функції **`Assert`**, **`retract`** й **`modify`** - три робітники конячки, використовувані більшістю правил.

Функція **`assert`** дозволяє додавати факти в список фактів поточної бази знань. Кожним викликом цієї функції можна додати довільне число фактів. У випадку якщо був включений режим перегляду зміни списку фактів, те відповідне інформаційне повідомлення буде відображатися у вікні CLIPS при додаванні кожного факту.

Визначення 10.5. Синтаксис

команди assert

```
(assert                    <факт>+)
```

При використанні команди **assert** необхідно пам'ятати, що перше поле факту обов'язково повинне бути значенням типу **symbol**. У випадку вдалого додавання фактів у базу знань, функція повертає адресу останнього доданого факту. Якщо під час додавання деякого факту відбулася помилка, команда припиняє свою роботу й повертає значення **FALSE**.

Слотам неупорядкованого факту, значення яких не задані, будуть привласнені значення за замовчуванням.

:

Приклад 10.8. Використання функції assert

```
(clear)
(assert (color red))
(assert (color blue)
     (value (+ 3 4)))
(deftemplate status
     (slot temp)
     (slot pressure
     (default low)))
(assert (status (temp high)))
```

Команда **clear** очищає поточний список фактів (а також всі певні конструктори, які вже були й ще буде розглянуті нижче). На відміну від **reset**, команда **clear** не додає в список фактів **initial-fact**. Цю команду також можна виконати, вибравши пункт **Clear CLIPS** у меню **Execution.** При виборі даної команди на екрані з'являється діалогове вікно, представлене на мал. 10.11. Це вікно запитує підтвердження користувача на очищення поточної бази знань.

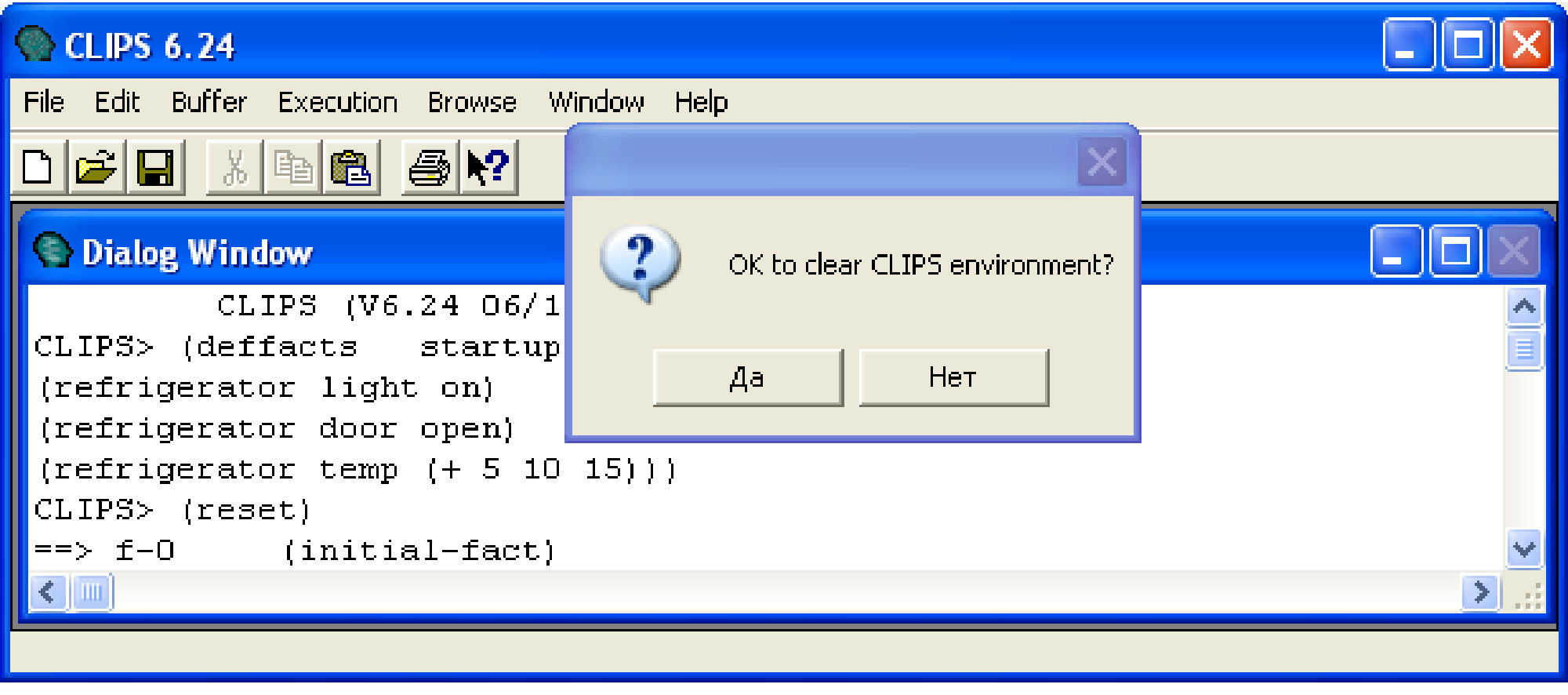

Рис. 10.11. Підтвердження очищення середовища CLIPS

У випадку, якщо команда була набрана із клавіатури, ніякого підтвердження на виконання цієї операції система не запитує. Якщо ви недавно почали працювати в середовищі CLIPS, то для очищення системи краще використати меню, тому що втрата всіх поточних даних з бази знань може виявитися досить хворобливою.

Включите режим перегляду зміни списку фактів і наберіть наведений вище приклад. Після цього виконаєте команду `(facts)`. Якщо при виконанні цих дій не було допущено помилок, то ви повинні одержати результат, ідентичний зображеному на мал. 10.12.

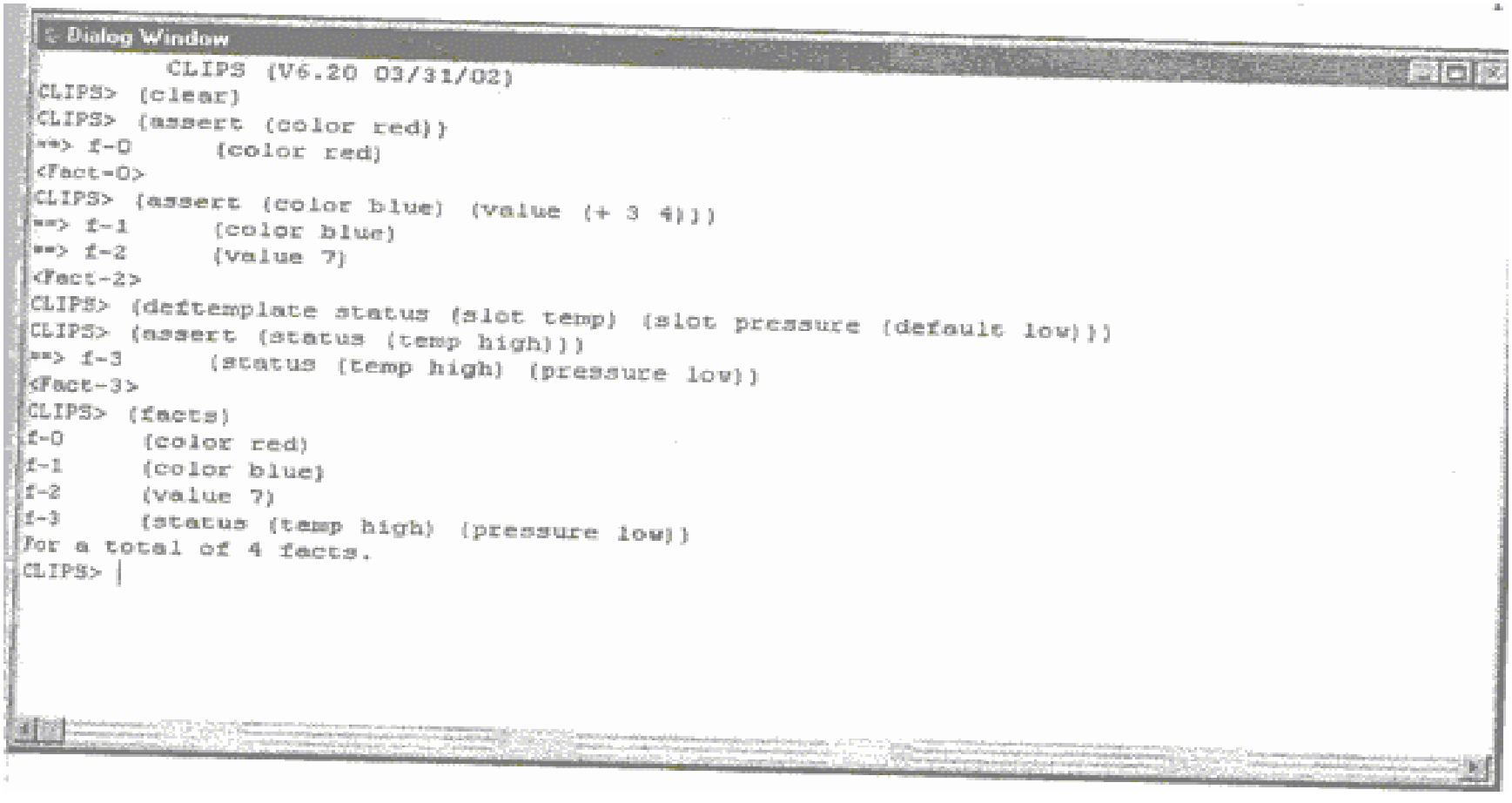

Рис. 10.12. Додавання фактів

Зверніть увагу, що при ініціалізації факту **value** використалося вираз, а слот **pressure** неупорядкованого факту **status** одержав значення за замовчуванням **low**.

За замовчуванням CLIPS не дозволяє додавати в список фактів два однакових факти. Наприклад, спроба додати два факти **color red** приведе до помилки й функція **assert** поверне значення **FALSE**. Дану установку системи можна змінити за допомогою функції **set-fact-duplication**. Крім того, користувачам Windows-версії CLIPS доступний ще один спосіб настроювання. Для цього необхідно відкрити діалогове вікно **Execution Options**, вибравши пункт **Options** з меню **Execution**, установити прапорець **Fact Duplication**. Зовнішній вигляд цього діалогового вікна наведений на мал. 10.6.

Функція *retract*

Після додавання факту в базу знань рано або пізно встане питання про те, як його відтіля видалити. Для видалення фактів з поточного списку фактів у системі CLIPS передбачена функція **`retract`**. Кожним викликом цієї функції можна видалити довільне число фактів. Видалення деякого факту може стати причиною видалення інших фактів, які логічно пов'язані із що видаляє. Крім того, видалення факту викликає видалення правил із ***плану рішення поточної задачі***, активованих видаляють фактом, що, але про це мова йтиме в наступних главах. У випадку якщо був включений режим перегляду зміни списку фактів, то відповідне інформаційне повідомлення буде відображатися у вікні CLIPS при видаленні кожного факту.

Визначення 10.6. Синтаксис команди retract

```
(retract <визначення-факту>+   \   *)
```

Аргумент <визначення-факту> може бути або змінної, пов'язаної з адресою факту за допомогою правила (ця можливість буде описана в наступній лекції), або індексом факту без префікса (наприклад, 3 для факту з індексом **f-3**), або вираженням, що обчислює цей індекс (наприклад, **(+ 1 2)** для факту з індексом **f-3**). Якщо як аргумент функції **retract** використався символ *****, то з поточної бази знань системи будуть вилучені всі факти. Функція **retract** не має значення, що повертає.

Для демонстрації роботи функції **retract** скористаємося ще одним візуальним інструментом, не описаним раніше. Він призначений для перегляду вмісту списку фактів у реальному часі. Цей інструмент доступний тільки користувачам Windows-версії системи CLIPS. Для того щоб активізувати перегляд списку фактів, поставте прапорець поруч із пунктом **Facts Window** меню **Windows**, як показано на мал. 10.13. Зовнішній вигляд інструмента перегляду списку фактів показаний на тім же малюнку. Відразу після запуску CLIPS цей список порожній.

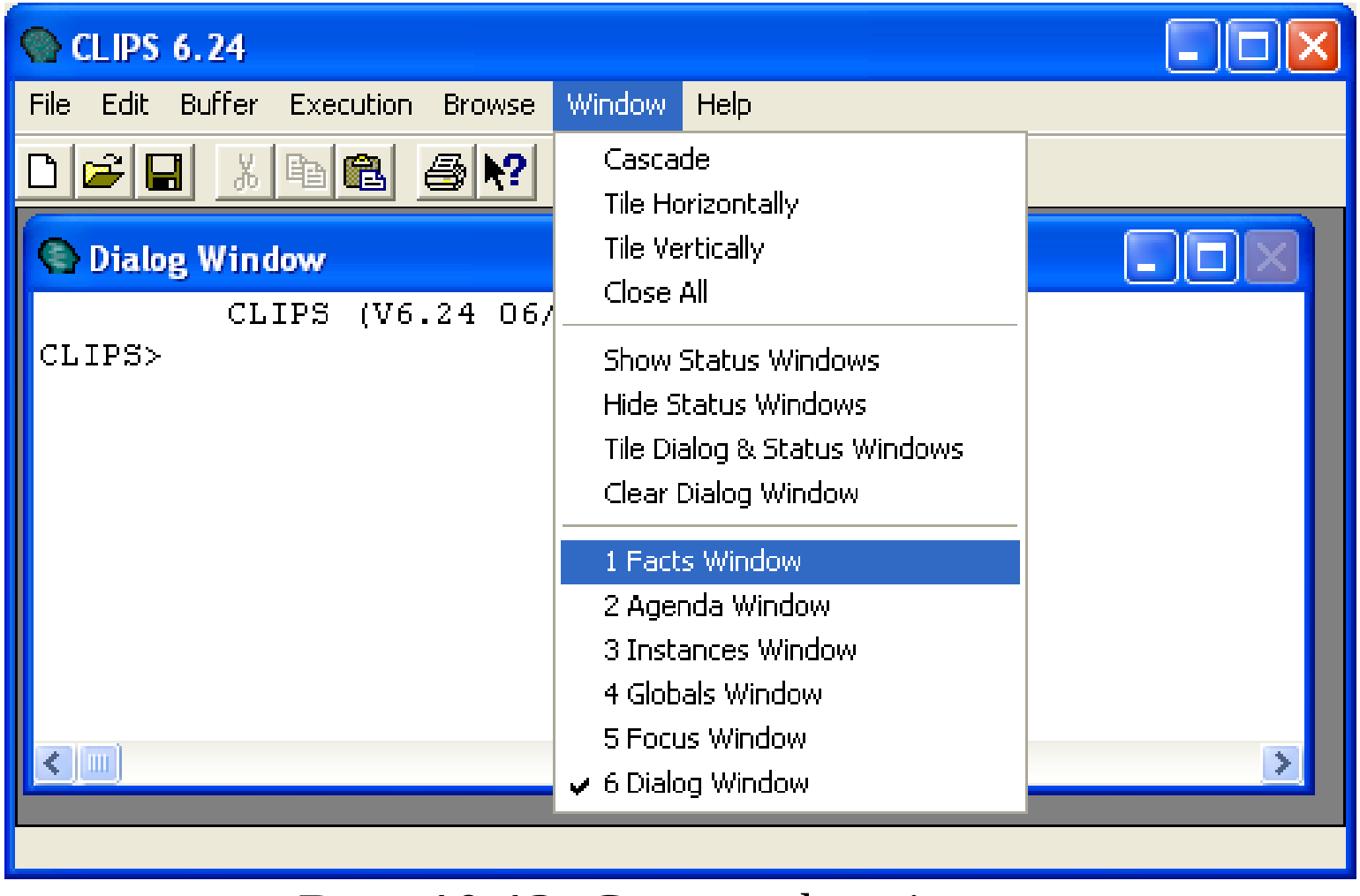

Рис. 10.13. Список фактів

Включите режим перегляду зміни списку фактів за допомогою діалогового вікна **Watch Options** і додайте в список фактів наступні факти:

Приклад 10.9. Додавання фактів

```
(assert  (a) (b) (c) (d) (e) (f))
```

Зверніть увагу, що у вікні перегляду фактів тепер відображаються всі 6 доданих фактів.

Приклад 10.10. Видалення фактів

```
(retract  0   (+  0  2)    (+ 0 2 2))
```

Ця команда видалить всі факти з парними індексами, використовуючи індекс факту безпосередньо (перший аргумент) і вираження, що обчислює індекс факту (другий і третій аргумент). Якщо перераховані команди були виконані правильно, то результат повинен відповідати мал. 10.14.

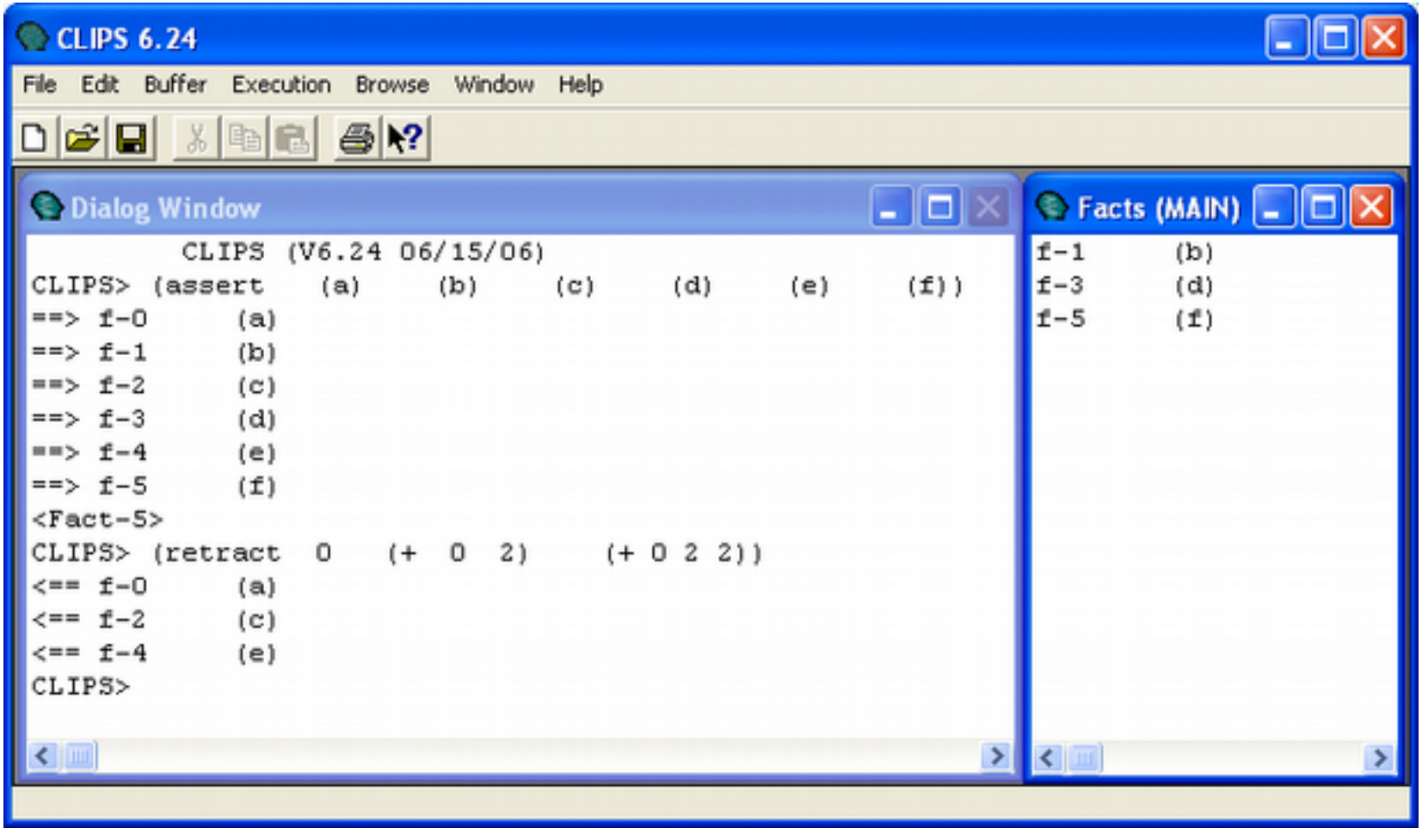

Рис. 10.14. Результат додавання й видалення фактів

У випадку, якщо факт із зазначеним індексом не буде знайдений, CLIPS видасть відповідне повідомлення про помилку.

Виконаємо команду:

Приклад 10.11. Видалення всіх фактів

```
(retract *)
```

Після виконання даної команди список фактів буде очищене повністю й вікно відображення поточного стану списку фактів стане ідентично зображеному на мал. 10.13.

Необхідно помітити, що функція retract не робить ніякого впливу на індекс наступних доданих фактів, тобто цей індекс не обнулюється. Якщо після видалення всіх уведених фактів додати в систему який-небудь факт, то він одержить індекс **`f-6`**, незважаючи на те, що список фактів у цей момент порожній.

Функція *modify*

Використовуючи функції assert й retract, можна виконувати більшість необхідних для функціонування правил дій. У тому числі й зміни існуючого факту. Наприклад, якщо в список фактів раніше був доданий факт (temperature is low), що одержав індекс 0, то змінити його значення можна, наприклад, у такий спосіб:

Приклад 10.12. Зміна існуючого факту

```
(clear)
(assert   (temperature is  low) )
(retract  0)
(assert   (temperature is high) )
```

Для зміни впорядкованих фактів доступний тільки цей спосіб. Для спрощення операції зміни неупорядкованих фактів CLIPS надає функцію **`modify`**, що дозволяє змінювати значення слотів таких фактів. **`Modify`** просто спрощує процес зміни факту, але її внутрішня реалізація еквівалентна викликам пар функцій **`retract`** й **`assert`**. За один виклик **`modify`** дозволяє змінювати тільки один факт. У випадку вдалого виконання функція повертає новий індекс модифікованого факту. Якщо в процесі виконання відбулася яка-небудь помилка, то користувачеві виводиться відповідне попередження й функція повертає значення FALSE.

Визначення 10.7. Синтаксис команди modify

```
(modify        <визначення-факту>
<новое-значение-слота>+)
```

Аргументом **<визначення-факту>** може бути або змінна, пов'язана з адресою факту за допомогою правила, або індекс факту без префікса (наприклад, 3 для факту з індексом **f-3**). Після визначення факту треба список з одного або більше нових значень слотів зазначеного шаблона. Для використання наведеного вище приклада його необхідно переробити в такий спосіб:

Приклад 10.13. Зміна існуючого неупорядкованого факту

```
(deftemplate temperature   (slot value) )
(assert   (temperature   (value low)) )
(modify 0   (value high) )
```

Якщо включити режим перегляду зміни списку фактів і виконати наведені вище команди, те отриманий результат повинен відповідати мал. 10.15.

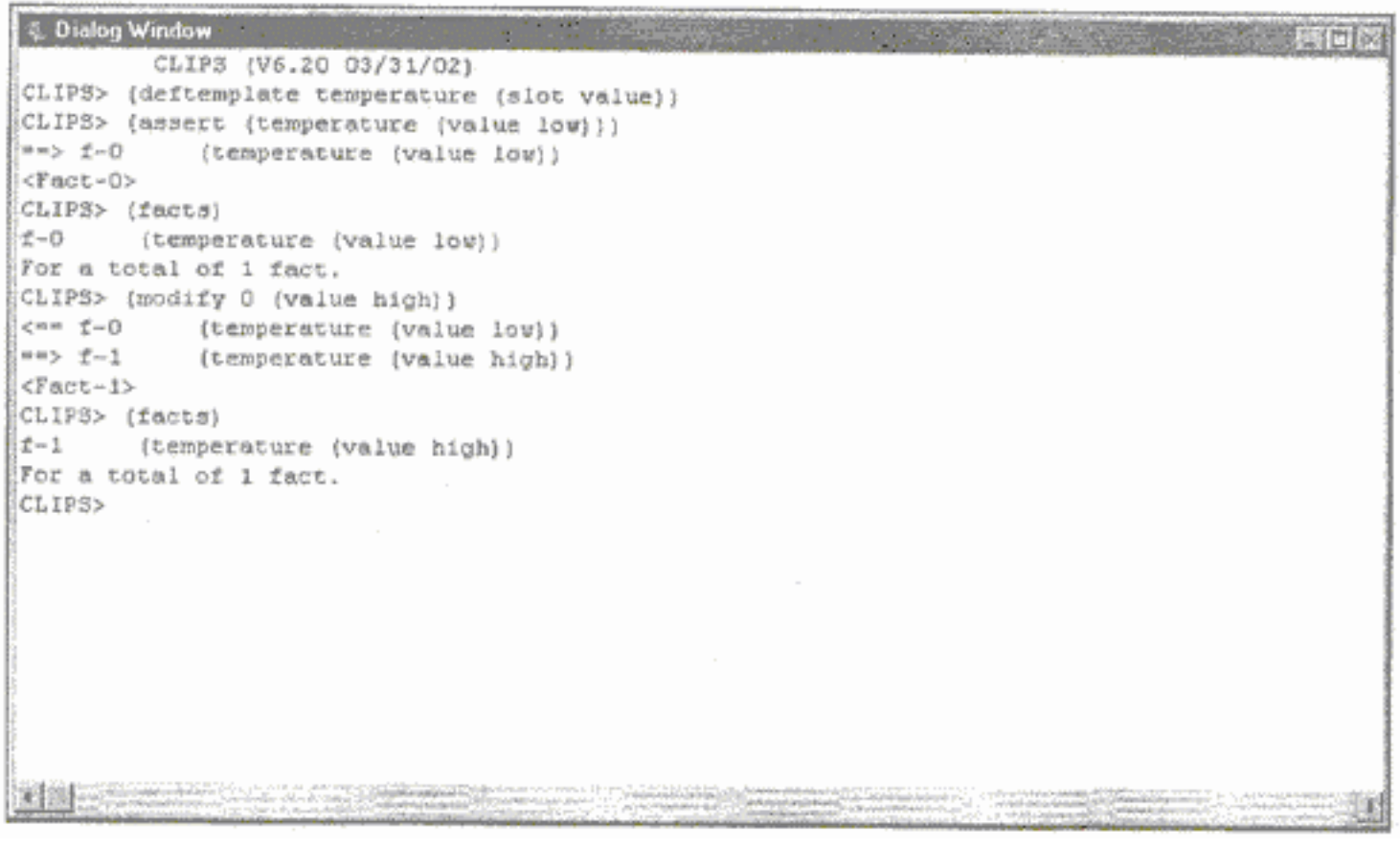

Рис. 10.15. Результат зміни існуючого неупорядкованого факту

Зверніть увагу на рух фактів у базі знань CLIPS при виконанні функції **`modify`** - спочатку віддаляється старий факт із індексом **`f-0`**, а потім додається новий факт із індексом **`f-1`**, ідентичний попередній, але з новим значенням заданого слоту.

Якщо в шаблоні заданого факту відсутній слот, значення якого потрібно змінити, CLIPS виведе відповідне повідомлення про помилку. Якщо заданий факт відсутній у списку фактів, користувач також одержить відповідне попередження.

Функція *duplicate*

Крім функції **modify**, в CLIPS існує ще одна дуже корисна функція, що спрощує роботу з фактами, — функція **duplicate**. Ця функція створює новий неупорядкований факт заданого шаблона й копіює в нього певну користувачем групу полів уже існуючого факту того ж шаблона. По діях, які виконує функція **duplicate**, аналогічна **modify**, за винятком того, що вона не видаляє старий факт зі списку фактів. Одним викликом функції **duplicate** можна створити одну копію деякого заданого факту. Як і функція **modify**, **duplicate**, у випадку вдалого виконання, повертає індекс нового факту, а у випадку невдачі — значення **FALSE**.

Визначення 10.8. Синтаксис команди duplicate

```
(duplicate     <визначення-факту>
               <новое-значение-слота>+)
```

Аргумент **<визначення-факту>** може бути або змінної, пов'язаної з адресою факту за допомогою правила, або індексом факту без префікса. Після визначення факту треба список з одного або більше нових значень слотів зазначеного шаблона. Продемонструємо роботу даної функції на наступному прикладі:

Приклад 10.14. Створення копії існуючого неупорядкованого факту

```
(deftemplate car
     (slot name)
     (slot producer)
     (slot type)
     (slot max-speed))
(assert ( car
     (name scorpio)
     (producer ford)
     (type sedan)
     (max-speed 180)))
```

```
(duplicate 0
     (type off-road)
     (max-speed 130))
```

У наведеному прикладі визначається шаблон, що описує властивості автомобіля, і додається факт - автомобіль Ford Scorpio з типом кузова седан і максимальна швидкість 180 (км/ч). Після цього за допомогою функції duplicate додається факт із інформацією про ще один автомобіль зі схожими характеристиками - це позашляховик Ford Scorpio з максимальною швидкістю 130 (км/ч). Duplicate просто полегшує нам життя, рятуючи від зайвого уведення значень даних співпадаючих слотів.

У випадку, якщо додає з допомогою **`duplicate`** факт уже присутній у списку фактів, буде видана відповідна інформація про помилку й повернуте значення **`FALSE`**. Факт при цьому доданий не буде. Це поводження можна змінити, дозволивши існування однакових фактів у базі знань.

Функція *assert-string*

Крім функції **`assert`**, CLIPS надає ще одну функцію, корисну при додаванні фактів, — **`assert-string`**. Ця функція приймає як єдиний аргумент символьний рядок, що є текстовим поданням факту (у тім виді, у якому ви набираєте його, наприклад, у функції **`assert`**), і додає його в список фактів. Функція **`assert-string`** може працювати як з упорядкованими, так і з неупорядкованими фактами. Одним викликом функції **`assert-string`** можна додати тільки один факт.

Визначення 10.9. Синтаксис команди assert-string

```
(assert-string          <строковий-вираз>)
```

Строкове вираження повинне бути укладене в лапки. Функція перетворить задане строкове вираження у факт CLIPS, розділяючи окремі слова на поля, з обліком певних у системі на сучасний момент шаблонів. Якщо в рядку необхідно записати внутрішнє строкове вираження, що представляє, скажемо, деяке поле, то для включення в строкове вираження символу лапок використається *зворотна коса риса* (backslash).

Наприклад, факт `(book-name "CLIPS user Guide")` можна додати в такий спосіб:

Приклад 10.15. Використання лапок усередині рядка

```
(assert-string "(book-name V'CLIPS User Guide\")")
```

Для додавання, що втримується в поле символу зворотної косої риси використайте неї двічі. Якщо зворотна коса повинна втримуватися усередині підрядку, її необхідно використати чотири рази.

Наприклад, для приміщення в поточний список факту **(a\b "c\d")** необхідно викликати функцію **assert-string** з наступним строковим аргументом:

Приклад 10.16. Використання зворотної косої риси

```
(assert-string "(a\\b \"c\\\\d\")")
```

Якщо додавання факту пройшло вдало, функція повертає індекс тільки що доданого факту, у противному випадку функція повертає повідомлення про помилку й значення **FALSE**. Функція **assert-string** не дозволяє додавати факт у випадку, якщо такий факт уже присутній *у* базі знань (якщо ви ще не включили можливість присутності однакових фактів).

Функція *fact-existp*

У цьому розділі розглянемо дуже просту, але надзвичайно важливу функцію **fact-existp**. Ця функція визначає, є присутнім чи в цей момент факт, заданий індексом або змінної покажчиком, у базі знань системи. У випадку якщо факт присутній у списку фактів, функція повертає значення **TRUE**, інакше — **FALSE**.

Визначення 10.10. Синтаксис команди fact-existp

```
( fact-existp      <визначення-факту>)
```

Звичайно ця функція застосовується в правилах, описаних у наступній лекції. Приведемо простий приклад використання даної функції:

Приклад 10.17. Використання функції fact-existp

```
(clear)
(assert-string  "(a\\b  \"c\\\\d\")")
(fact-existp  0)
(retract  0)
(fact-existp  0)
```

Зауваження

Не забудьте виконати команду **`clear`**, щоб доданий факт мав нульовий індекс. Після першого виклику функція **`fact-exist`** поверне значення **`TRUE`**, а після видалення факту `3` індексом 0 — **`FALSE`**.

Функції для роботи з неупорядкованими фактами

Для роботи з неупорядкованими фактами в CLIPS передбачений цілий ряд спеціальних функцій. До них ставляться: **`fact-relation`**, **`fact-slot-names`** й **`fact-slot-value`**. Розглянемо ці функції одну за одною.

Функція **`fact-relation`** дозволяє одержати *зв'язок* (relation) існуючого факту із шаблоном. Зв'язок факту із шаблоном, певним за допомогою конструктора **`deftemplate`** або неявно створеним шаблоном, визначається по першому полю факту. Це поле завжди є простим полем і використається CLIPS як ім'я шаблона, з яким зв'язаний факт. Таким чином, функція **`fact-relation`** просто повертає перше поле факту, або значення **`FALSE`**, якщо зазначений факт не знайдений.

Визначення 10.11. Синтаксис команди fact-relation

```
(fact-relation      <визначення-факту>)
```

Як визначення факту, як й в описані вище функціях, потрібно використати або змінну покажчик, що містить адресу факту, або індекс факту.

Приклад 10.18. Використання функції fact-relation

```
(clear)
(assert (car Ford))
(fact-relation 0)
(retract 0)
(fact-relation 0)
```

У першому випадку функція **fact-relation** поверне значення **car**, а в другому - **FALSE**.

Для одержання імен всіх слотів заданого факту в CLIPS призначена функція **fact-slot-names**.

Визначення 10.12. Синтаксис команди fact-slot-names

```
(fact- slot-names        <визначення-факту>)
```

Дана функція повертає список імен слотів у складеному полі. Для впорядкованих фактів функція повертає значення **implied** (який мається на увазі), тому що, якщо ви помнете, CLIPS представляє впорядковані факти як неявно задані неупорядковані з одним складеним слотом. У випадку якщо заданий факт не знайдений, функція повертає значення **FALSE**.

Приклад 10.19. Використання функції fact-slot-names

```
(clear)
(deftemplate  car
     (slot name)
     (slot producer)
     (slot type)
     (slot max-speed))
(assert (   car
     (name scorpio)
     (producer ford)
     (type sedan)
     (max-speed 180)))
(fact-slot-names 0)
```

Якщо наведений приклад був набраний без помилок, то функція **`fact-slot-names`** поверне значення **`(name producer type max-speed)`**.

Останньою з розглянутих у даній лекції функцій для роботи з неупорядкованими фактами буде функція **`fact-slot-value`**.

Визначення 10.13. Синтаксис команди fact-slot-value

```
(fact-slot-value    <визначення-факту>
<имя-слота  >)
```

Дана функція дозволяє одержувати значення слоту деякого заданого факту. Якщо факт є впорядкованим, то для одержання значення неявно певного складеного слоту використається значення **IMPLIED**. У випадку якщо зазначений факт не існує, або ім'я слоту зазначене не вірно, функція повертає значення **FALSE**.

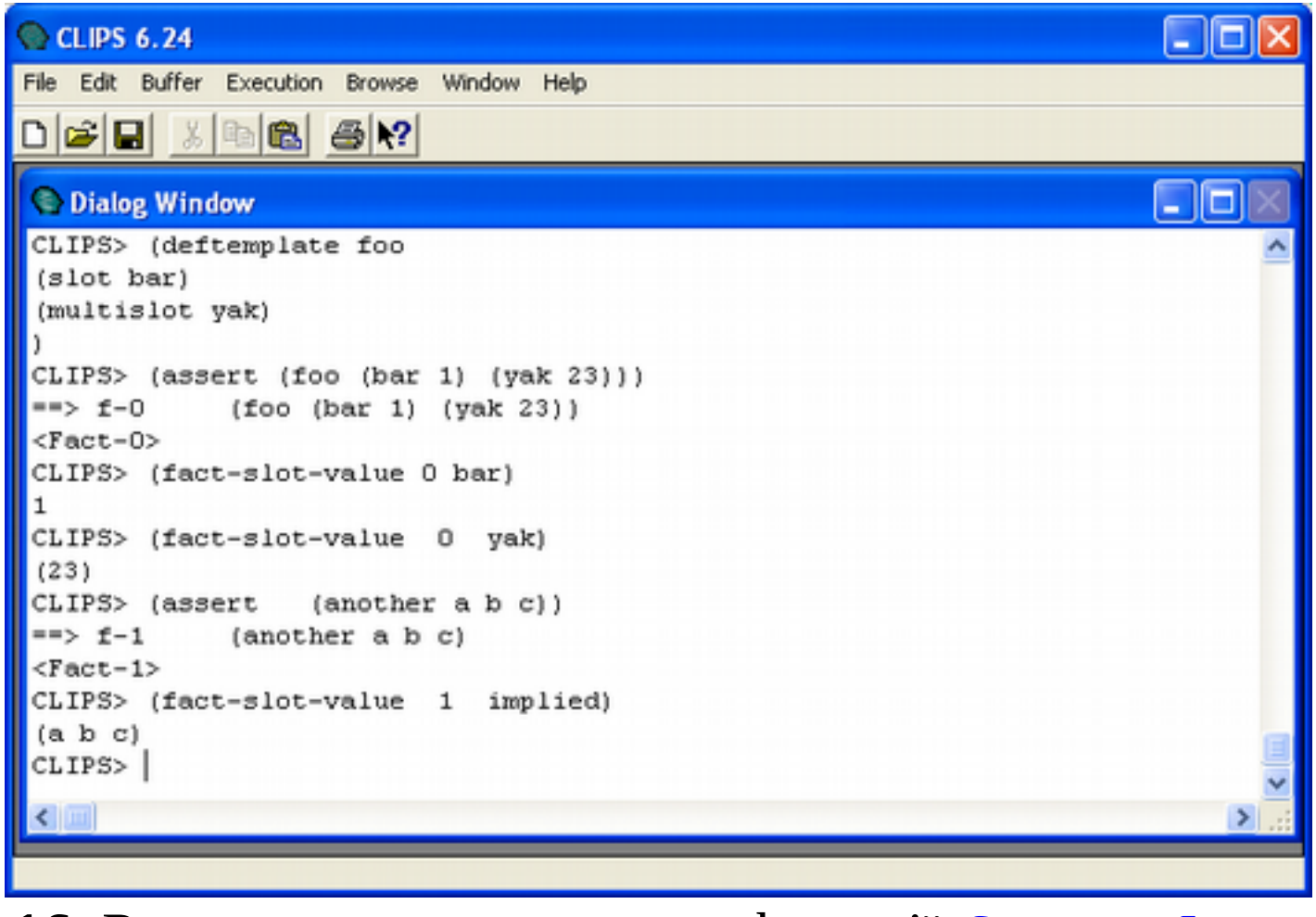

Рис. 10.16. Результат використання функції **`fact-slot-value`**

Виконаємо в середовищі CLIPS наступний приклад:

Приклад 10.20. Використання функції fact-slot-value

```
(clear)
(deftemplate foo
     (slot bar)
     (multislot yak)
(assert (foo (bar 1) (yak 23)))
(fact-slot-value 0 bar)
(fact-slot-value  0  yak)
(assert   (another a b c))
(fact-slot-value  1  implied)
```

Якщо попередній приклад був виконаний без помилок, то отриманий результат повинен відповідати наведеному на мал. 10.16.

Функції збереження й завантаження списку фактів

Як можна помітити, наповнення списку фактів в CLIPS досить кропітке й тривале заняття. Якщо фактів досить багато, цей процес може розтягтися на кілька годин, або навіть днів. Тому що список фактів зберігається в оперативній пам'яті комп'ютера, теоретично, через збій комп'ютера або, наприклад, несподіваного відключення харчування, список фактів можна безповоротно втратити. Щоб цього не відбулося, а так само для того щоб зробити роботу з наповнення бази знань фактами більше зручної, CLIPS надає команди збереження й завантаження списку фактів у файл - **`save-facts`** й **`load-facts`** відповідно.

Визначення 10.14. Синтаксис команди save-facts

```
(save-facts <імені-файлу> [<межі-видимості> <списків-шаблонів >])
<межі-видимості> ::= visible|local
```

Команда **save-facts** зберігає факти з поточного списку фактів у текстовий файл. На кожен факт приділяється один рядок. Неупорядковані факти зберігаються разом з іменами слотів. У функції існує можливість обмежити область видимості фактів, що зберігають. Для цього використається аргумент <межі-видимості>. Він може приймати значення **local** або **visible**. У випадку якщо цей аргумент приймає значення **visible**, те зберігаються всі факти, що є присутнім у цей момент у системі.

Якщо як аргумент використається ключове слово **`local`**, то зберігаються тільки факти з поточного модуля. Про модулі мова в наступних лекціях. За замовчуванням аргумент **`<межі-видимості>`** приймає значення **`local`**. Після аргументу **`<межі-видимості>`** може випливати список певних у системі шаблонів. У цьому випадку будуть збережені тільки ті факти, які пов'язані із зазначеними шаблонами.

Приклад 10.21. Використання функції save-facts

```
(clear)
(deftemplate template
(slot a)
(slot b))
(assert   (template   (a  1)     (b  2)))
(assert   (simple-factl)     (simple-fact2))
(save-facts  fl  local  template simple-factl)
```

Послідовність дій, наведена в даному прикладі, зберігає у файл **`f1`**, що перебуває в поточному каталозі, всі факти, видимі в поточному модулі й пов'язані із шаблонами **`template`** й **`simple-factl`** (як ви пам'ятаєте, після додавання факту **`simple-factl`** CLIPS визначає неявно створений шаблон **`simple-factl`**). У результаті буде отриманий текстовий файл із наступним змістом:

Приклад 10.22. Зміст файлу fl

```
(template   (a  1)    (b  2) } (simple-factl)
```

У випадку успішного виконання, команда повертає значення **TRUE**, а у випадку невдачі — відповідне повідомлення про помилку. Якщо зазначений файл уже існує, він буде перезаписаний.
Для завантаження збережених раніше файлів використається функція **LOAD-FACTS**. Функція має наступний формат:

Визначення 10.15. Синтаксис команди load-facts

```
(load-facts  <ім'я-файлу>)
```

Тут **<ім'я-файлу>** - ім'я текстового файлу, збереженого раніше за допомогою команди **save-facts**, що містить список фактів. Файл зі списком фактів можна також створити в будь-якому текстовому редакторі, якщо ви добре розібралися з поданням фактів в CLIPS. Для завантаження збереженого в попередньому прикладі файлу виконаєте:

Приклад 10.23. Використання функції load-facts

```
(load-facts     fl)
```

У випадку успішного виконання команда повертає значення **TRUE**, а у випадку невдачі — **FALSE** і відповідне повідомлення про помилку. Зверніть увагу, що якщо у файлі втримуються факти, пов'язані з явно створеними за допомогою конструктора **DEFTEMPLATE** шаблонами, то в момент завантаження всі необхідні шаблони повинні бути вже визначені в системі. Якщо ця умова не буде виконано, то завантаження фактів закінчиться невдало. На щастя, CLIPS також дозволяє й завантаження конструкторів з текстового файлу, але про це ми поговоримо в наступній главі, після розгляду конструктора **DEFRULE**.

Наступна лекція буде присвячена визначенню правил і системі CLIPS