МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №1
по дисциплине «Представление знаний в системах искусственного интеллекта»
Тема: Изучение основных возможностей и базовых команд среды CLIPS. Разработка демонстрационной экспертной системы

Студент гр. 1310 _____ Комаров Д.Е.

Преподаватель _____ Сучков А.И.

Санкт-Петербург
2025

Цель работы.

Целью выполнения лабораторной работы является изучение базовых команд и возможностей среды CLIPS, а также разработка на основе полученных навыков демонстрационной экспертной системы.

Основные теоретические положения.

Среда CLIPS (C Language Integrated Production System) разработана в начале 1980-х годов в Космическом центре Джонсона NASA и предназначена для построения ЭС и поддерживает три основных способа представления знаний:

- продукционные правила для представления эвристических знаний;
- функции для представления процедурных знаний;
- объектно-ориентированное программирование.

Данная среда имеет следующие базовые типы данных:

- целые (integer) и вещественные (float) числа;
- символьные (symbol) и строковые (string) значения;
- внешний адрес (external-address);
- адрес факта (fact-address);
- имя (instance-name) и адрес экземпляра (instance-address).

Факты — основная форма представления данных в CLIPS. Факты бывают упорядоченные и неупорядоченные.

Факты управляются командами: assert, retract, modify, duplicate. Исходные факты объявляются через deffacts и загружаются при reset.

Правила задаются конструкцией defrule. Правило активируется, если все условия (антецедент) выполняются. Тогда оно помещается в агенду и выполняется (консеквент).

Постановка задачи.

1. Изучение базовых команд и конструкций CLIPS.

1.1. Запустить систему CLIPS. Активизировать окно просмотра текущего списка фактов. Выполнить следующую последовательность действий, фиксируя после каждого шага состояние списка фактов:

1) сбросить систему в исходное состояние;

2) выполнить начальную установку;

3) ввести 3 любых упорядоченных факта;

4) повторно выполнить сброс;

5) установить 3 ранее вводимых упорядоченных факта в качестве исходных фактов;

6) выполнить сброс.

1.2. Активизировать дополнительно окно просмотра агенды. Выполнить следующую последовательность действий, фиксируя после каждого шага состояния списка фактов и агенды:

1) ввести три правила, такие, что антецеденты первых двух правил сопоставляются с комбинацией фактов, заданных ранее конструкцией (deffacts), а консеквентны этих правил добавляют новые факты, сопоставляемые с антецедентом третьего правила;

2) выполнить по шагам активизацию правил.

2. Разработка демонстрационной экспертной системы.

2.1. Выбрать (придумать) предметную область демонстрационной экспертной системы (ЭС). Сформировать, пользуясь редактором CLIPS, базу знаний демонстрационной ЭС для выбранной предметной области и сохранить ее в файле rulebase.clp.

2.2. Для активизации ЭС в среде CLIPS использовать пакетный файл Run_Lab_1.bat.

2.3. Оттестировать ЭС на различных комбинациях входных значений в пошаговом режиме. Представить отчет о практической работе с приложением исходного текста ЭС.

В команде, выполнявшей данную лабораторную работу, было следующее разделение ролей:

- Будаев Геннадий — изучение базовых команд и возможностей CLIPS, формирование концепции ЭС, написание кода демонстрационной ЭС, написание цели, основных теоретических положений и постановки задачи работы, тестирование ЭС;
- Комаров Дмитрий — изучение базовых команд и возможностей CLIPS, формирование концепции ЭС, написание кода демонстрационной ЭС, рисование схемы ЭС, написание 1 раздела отчета;
- Шейнов Кирилл — изучение базовых команд и возможностей CLIPS, формирование концепции ЭС, написание кода демонстрационной ЭС, написание 2 раздела отчета, формирование оформления ЭС.

Выполнение работы.

1. Изучение базовых команд и конструкций CLIPS.

Запустим среду CLIPS, сбросим ее в исходное состояние и выполним начальную установку. На рисунке 1 представлено окно среды после выполнения данных операций.

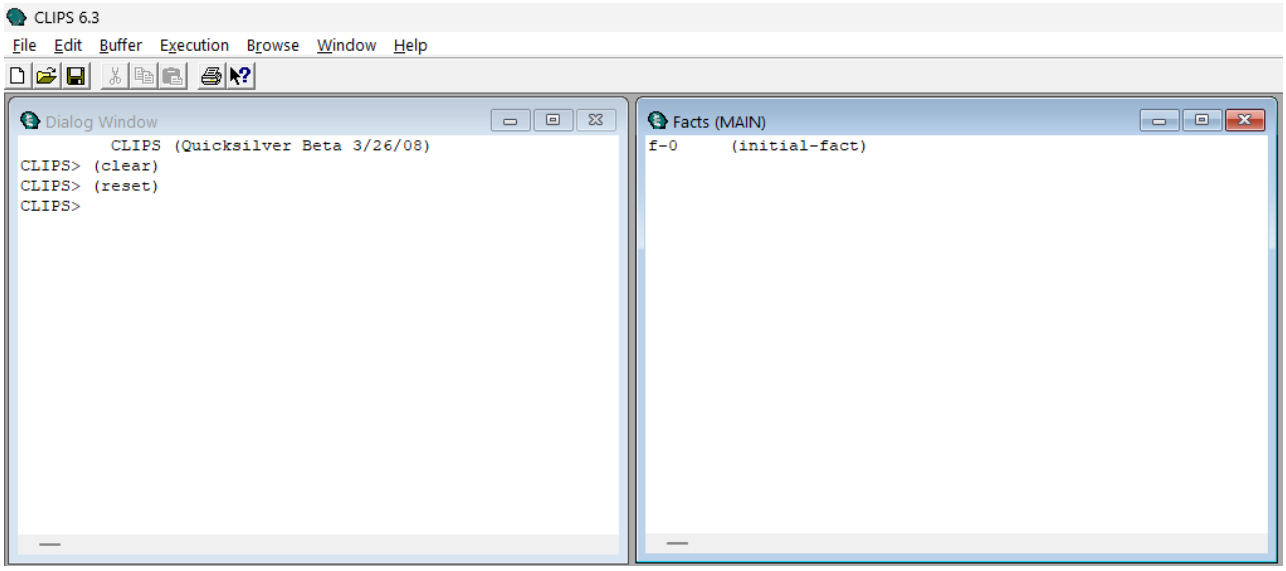

Рисунок 1 – Окно CLIPS после сброса в исходное состояние и выполнения начальной установки

Введем 3 любых упорядоченных факта, например (a a), (b b) и (c c). На рисунке 2 представлено окно CLIPS после ввода этих фактов.

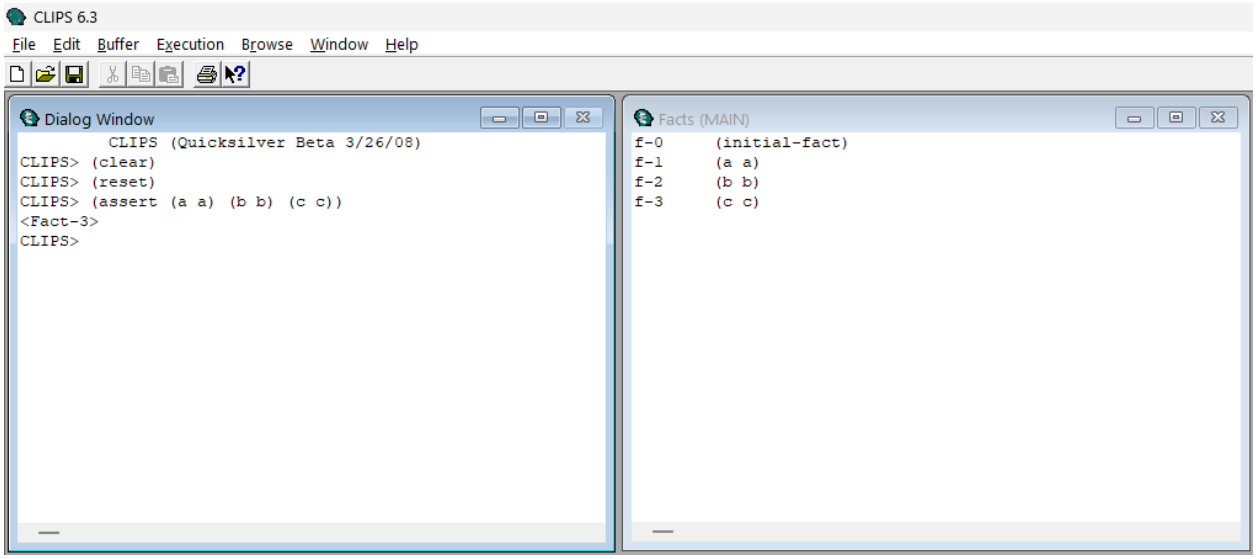

Рисунок 2 – Окно CLIPS после ввода трех фактов

На рисунке 2 видно, что ранее введенные в консоль факты отобразились в списке фактов. Выполним сброс повторно. На рисунке 3 представлено окно CLIPS после повторного сброса.

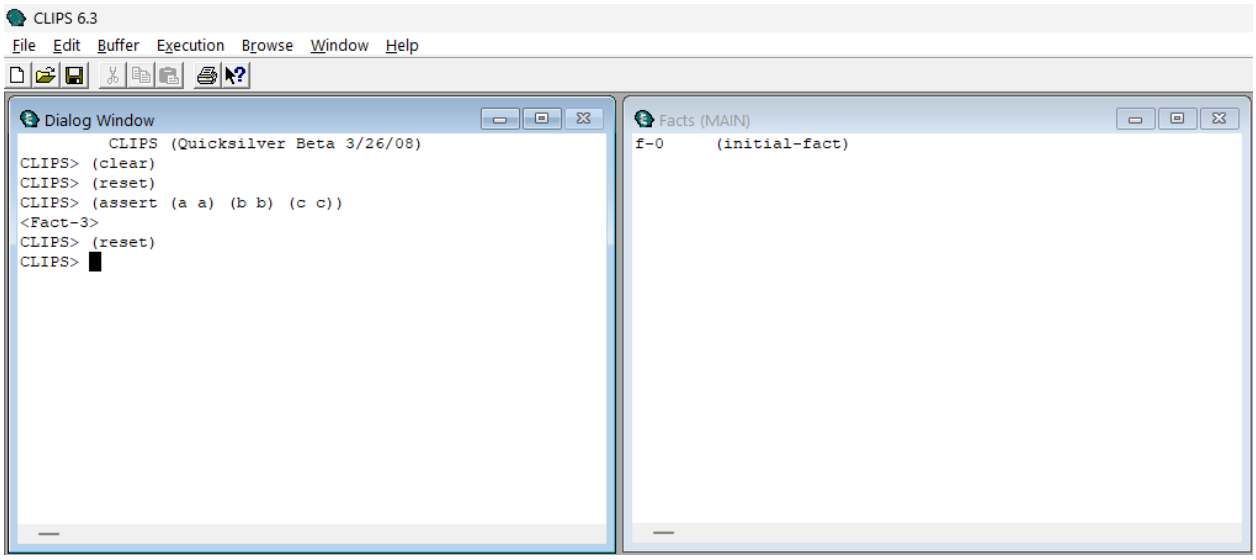

Рисунок 3 – Окно CLIPS после повторного сброса

Из рисунка 3 видно, что после сброса все ранее введенные в консоль факты были удалены из окна фактов. Установим те же самые факты в качестве исходных и выполняем сброс. На рисунке 4 представлено окно CLIPS после проведенных операций. Исходные факты появляются в окне фактов сразу после сброса.

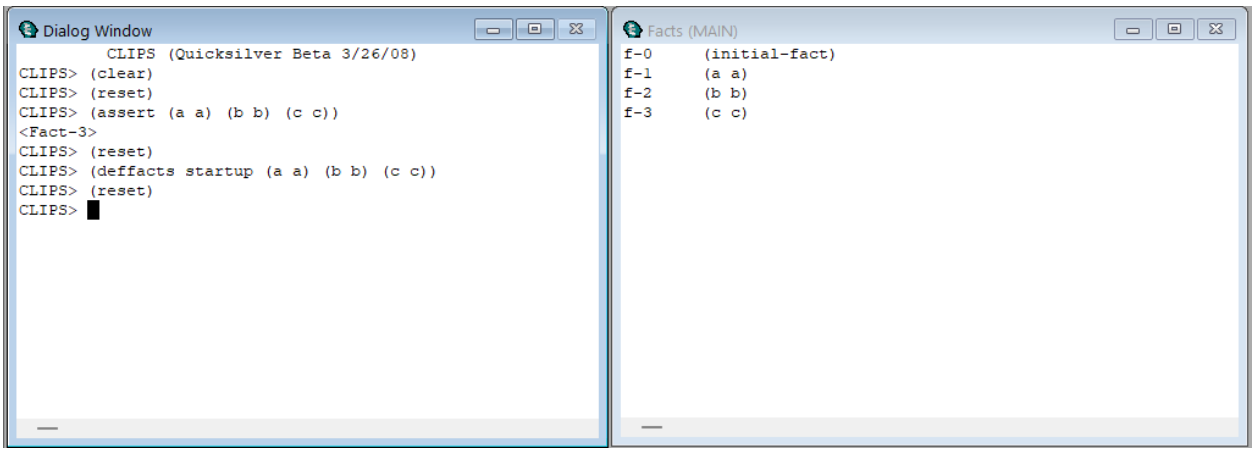

Рисунок 4 – Окно CLIPS введения исходных фактов и сброса

Откроем окно агенды и введем 3 правила. На рисунке 5 представлено окно CLIPS после проведенных операций.

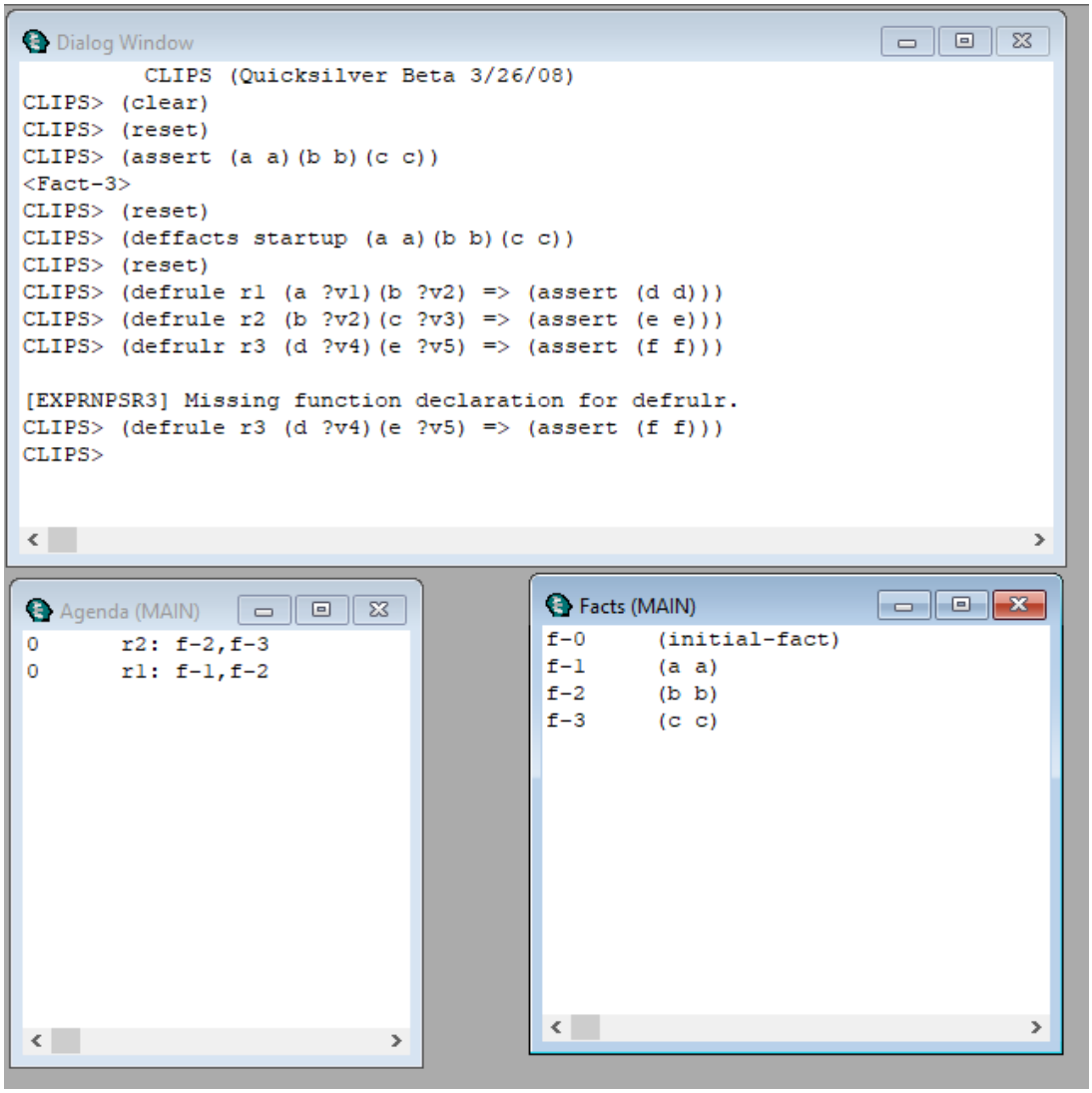

Рисунок 5 – Окно CLIPS после ввода трех правил

Как видно из окна агенты на рисунке 5, 2 правила уже могут активироваться. Выполним активацию правил по шагам. На рисунке 6 показано окно CLIPS после первого шага.

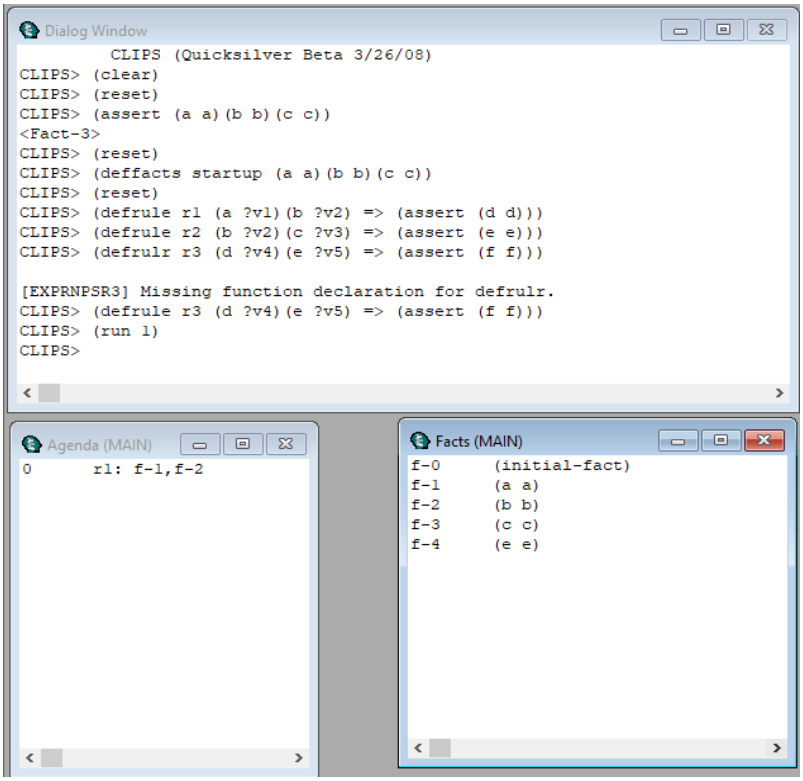

Рисунок 6 – Окно CLIPS после первого шага

Как можно заметить, первым активировалось правило r2 и добавило факт (e e), поскольку оно было введено позже, а правила работают по принципу стека. Выполним еще один шаг. На рисунке 7 показано окно CLIPS после второго шага.

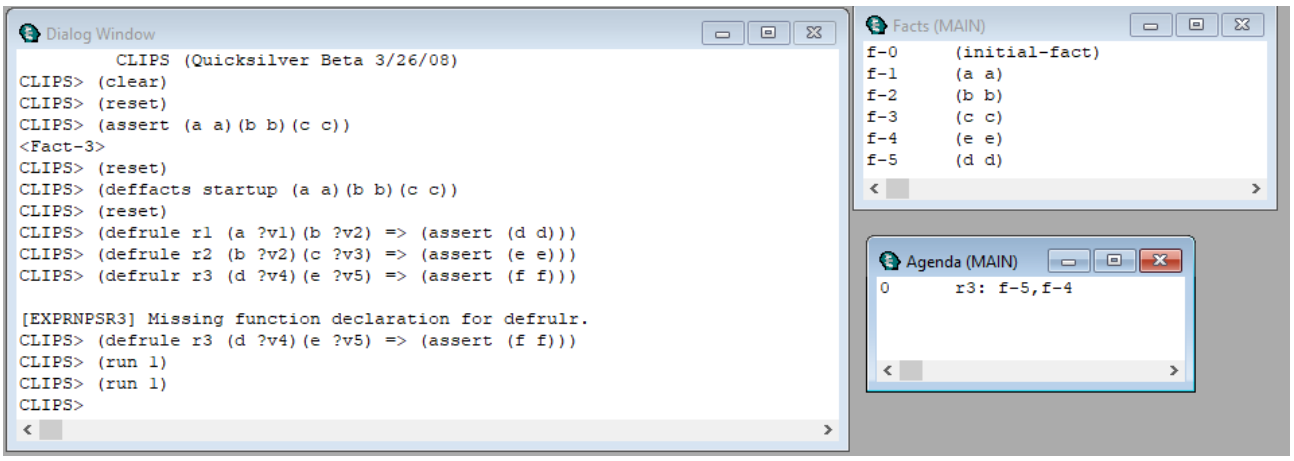

Рисунок 7 – Окно CLIPS после второго шага

После активации первого правила и добавления факта (d d), фактов стало достаточно, чтобы активировалось третье правило. На рисунке 8 показано окно CLIPS после третьего шага.

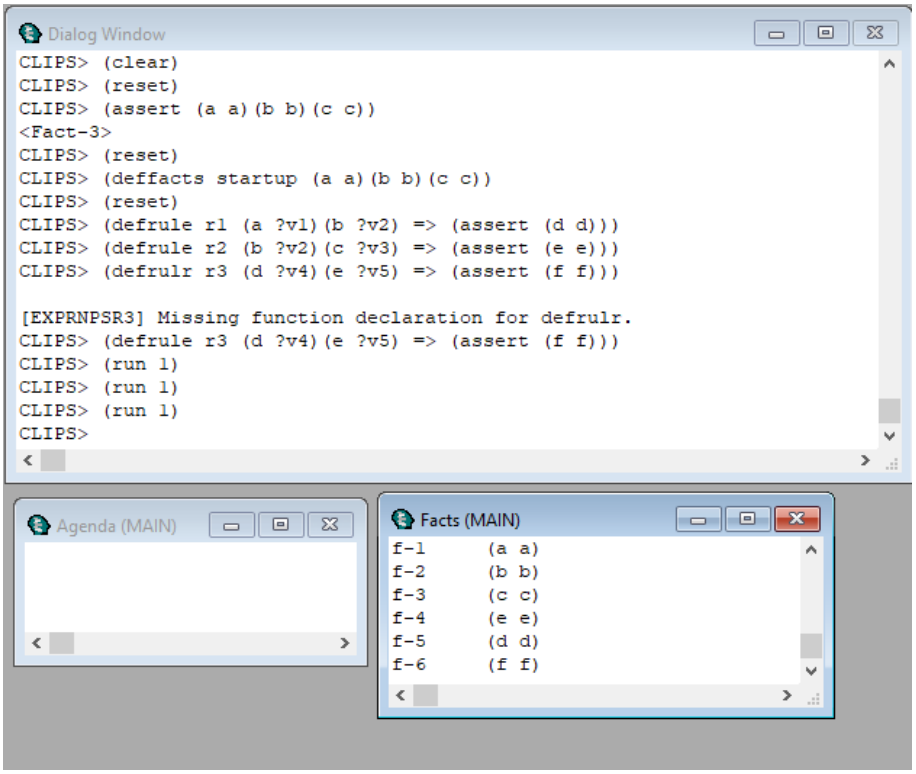

Рисунок 8 – Окно CLIPS после третьего шага

После активации всех правил окно агенды осталось пустым и был добавлен факт (f f).

2. Разработка демонстрационной экспертной системы.

В качестве предметной области для создания демонстрационной экспертной системы была выбрана диагностика неисправностей компьютера. Данная экспертная система помогает пользователю ПК самостоятельно локализовать неисправность своего устройства. Путем последовательных вопросов система принимает решение о том, какая неисправность наиболее вероятна и выдает пользователю советы по ее устранению. На рисунке 9 представлена схема получившейся экспертной системы.

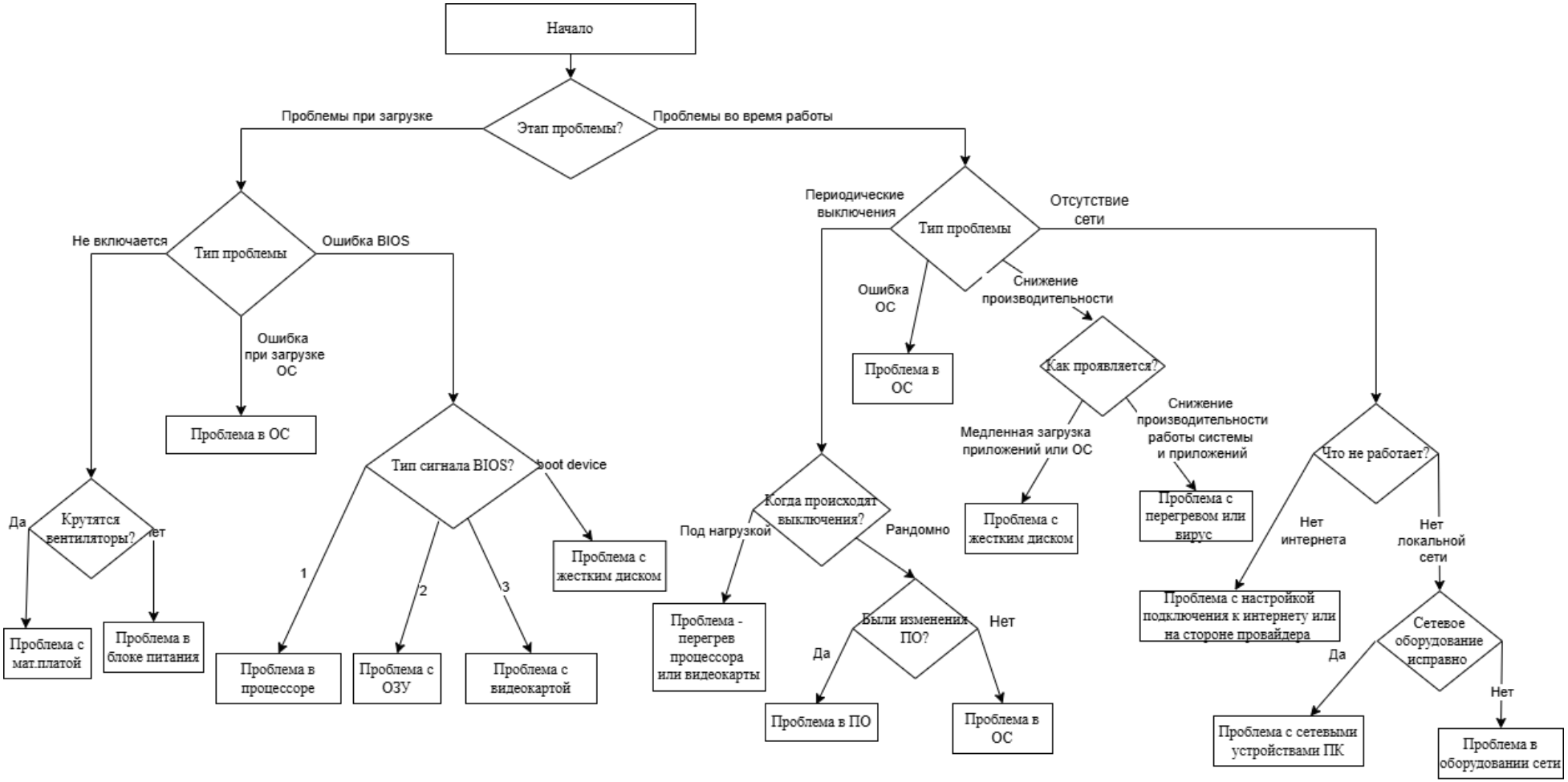

Рисунок 9 – Схема экспертной системы

Работа с экспертной системы в среде CLIPS начинается с загрузки программного файла программы в формате clp использованием пункта «Load» меню «File». После загрузки файла и сброса системы командой Reset в окне агенды отображается активное начальное правило, с которого ЭС начинает задавать вопросы пользователю. Окно CLIPS после загрузки ЭС и сброса представлено на рисунке 10.

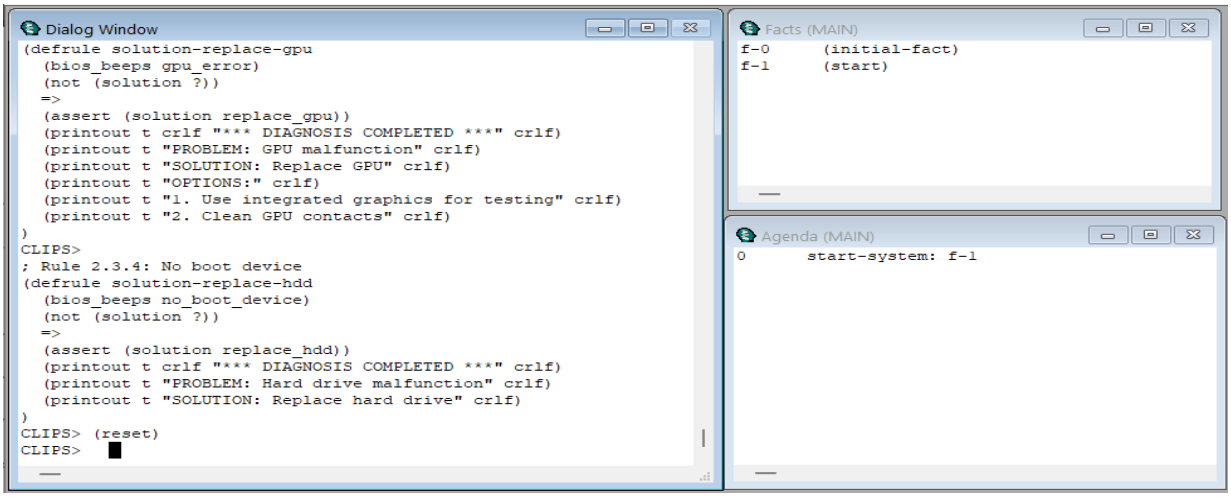

Рисунок 10 – Инициализация ЭС

После запуска ЭС в DialogWindow отображается стартовое сообщение с вопросом, на каком этапе возникает проблема с ПК – рисунок 11.

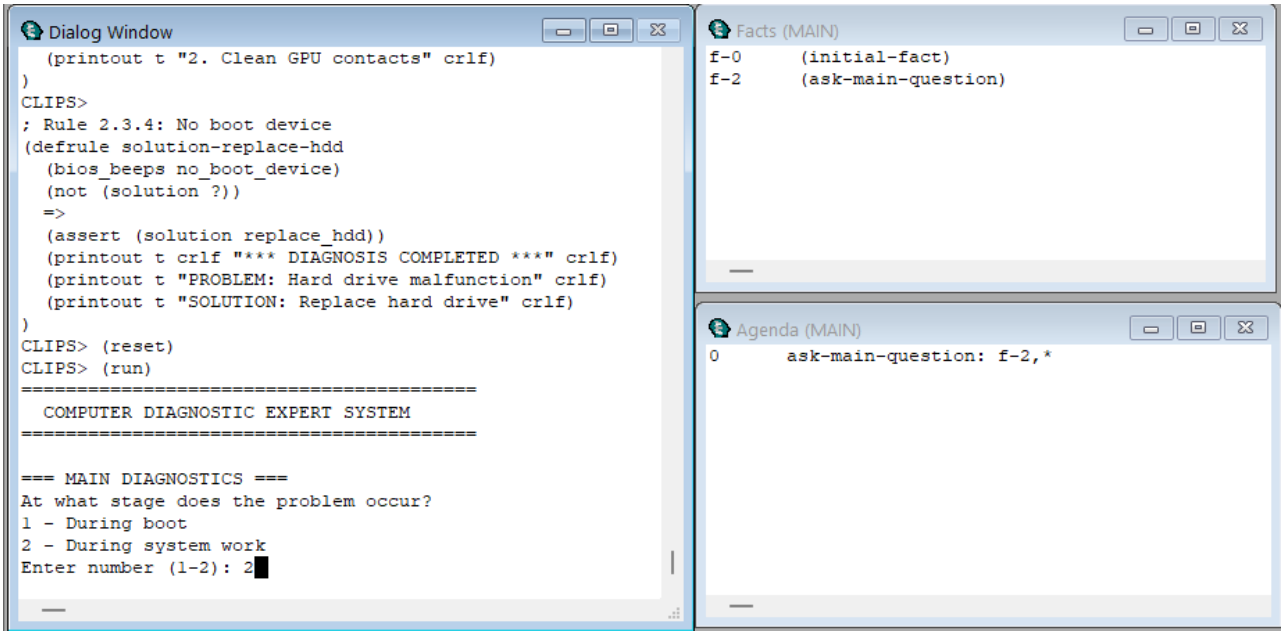

Рисунок 11 – Стартовое сообщение

Система обновляет список фактов, в зависимости от выбранного варианта контекста проявления проблемы с ПК, что отображено на рисунке 12.

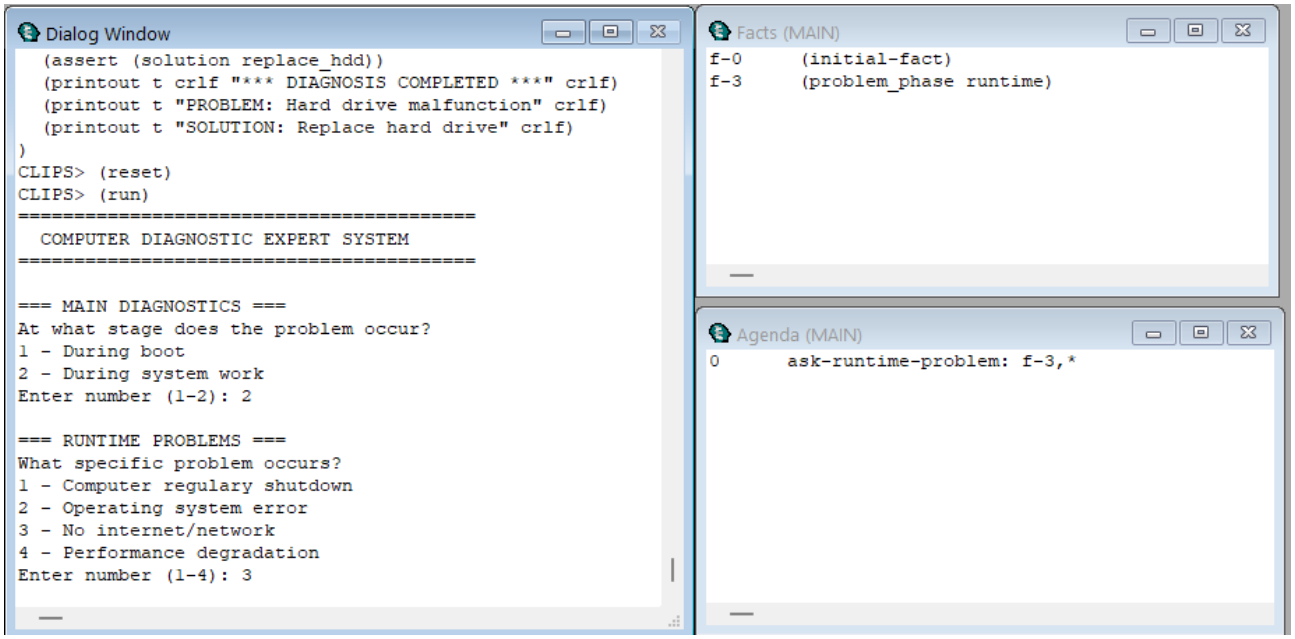

Рисунок 12 – Уточнение неисправности во время работы ПК

После этого происходит уточнение неисправности вопросами, которые позволяют локализовать проблему, что отображено на рисунках 13, 14. По ходу выполнения программы в окне фактов добавляются активные факты.

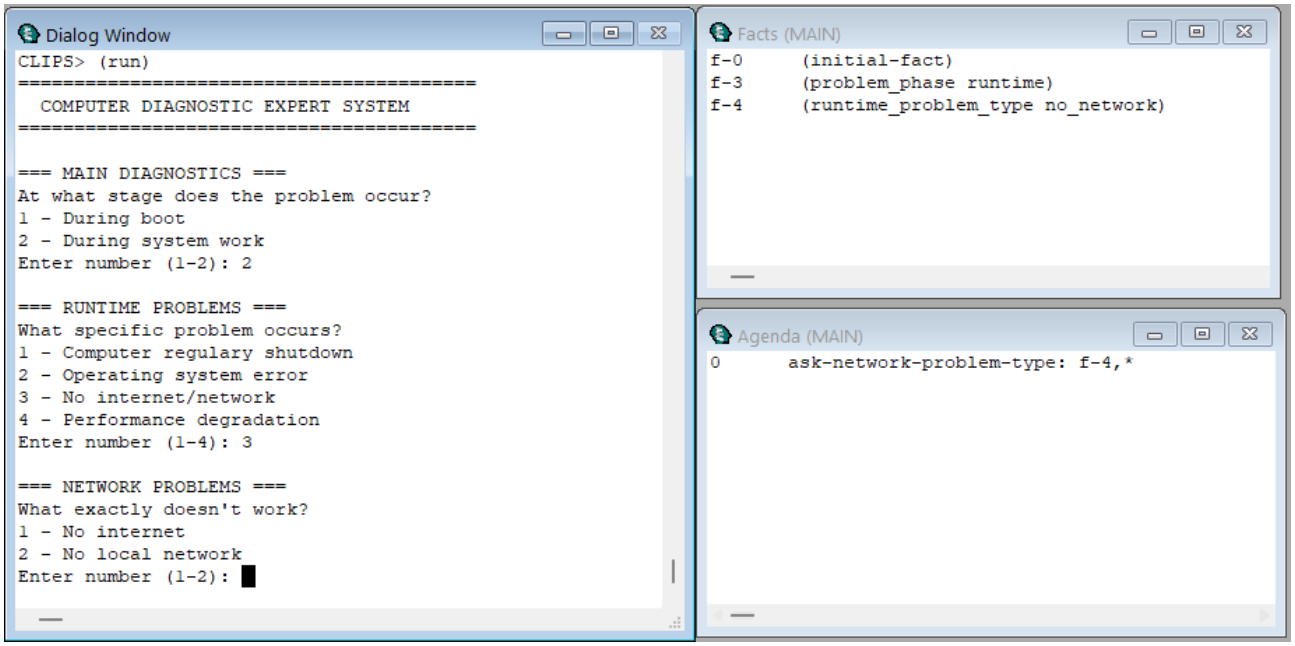

Рисунок 13 – Уточнение проблемы с сетевым подключением

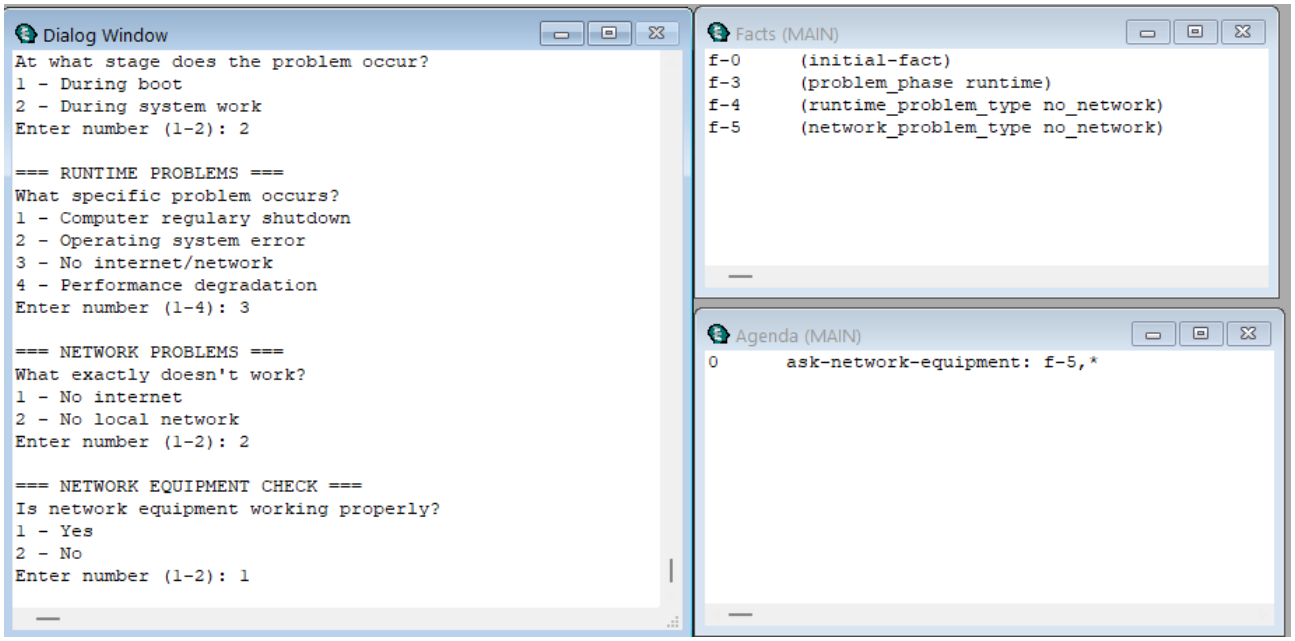

Рисунок 14 – Текущий вопрос исправности сетевого оборудования

При достижении локализации проблемы в DialogWindow выводится рекомендуемое решение проблемы с ПК исходя из введенных данных, так же оно отображается в окне фактов – рисунок 15.

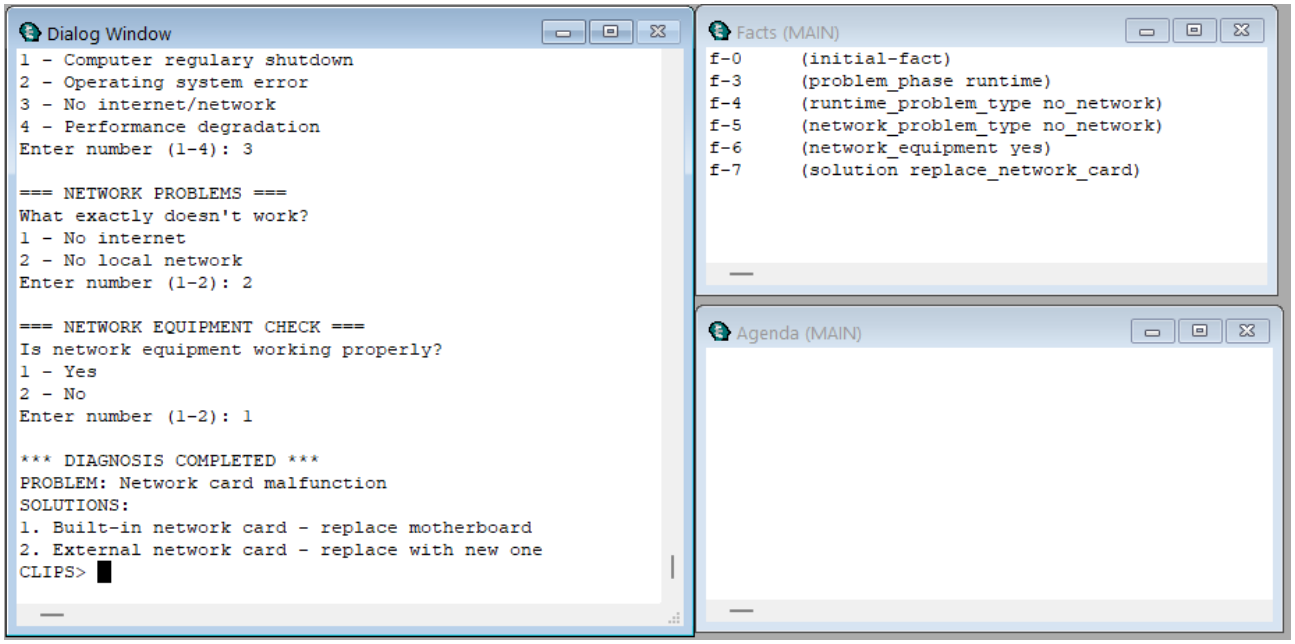

Рисунок 15 – Завершение диагностики

Исходный код экспертной системы на языке CLIPS представлен в приложении А.

Выводы.

В ходе выполнения лабораторной работы были изучены базовые команды и возможности среды CLIPS. Были изучены способы задания фактов и создания правил для их интерпретации. Также на основании полученных навыков была разработана демонстрационная экспертная система, помогающая пользователю самостоятельно диагностировать свой ПК.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ЭКСПЕРТНОЙ СИСТЕМЫ

```
; COMPUTER DIAGNOSTIC EXPERT SYSTEM
;-----------------------------------------------
; INPUT VALIDATION FUNCTION
;-----------------------------------------------
(deffunction ask-with-validation (?question $?allowed-values)
  (bind ?valid-input FALSE)
  (while (not ?valid-input)
    (printout t ?question)
    (bind ?answer (read))
    (if (member ?answer ?allowed-values)
      then
      (bind ?valid-input TRUE)
      (return ?answer)
      else
      (printout t "Invalid input! Allowed values: " ?allowed-
values crlf)
    )
  )
)

;-----------------------------------------------
; SYSTEM INITIALIZATION
;-----------------------------------------------

(deffacts initialization
  (start)
)

(defrule start-system
  ?f <- (start)
  =>
  (retract ?f)
  (printout t "========================================" crlf)
  (printout t "  COMPUTER DIAGNOSTIC EXPERT SYSTEM" crlf)
  (printout t "========================================" crlf)
  (assert (ask-main-question))
)

;-----------------------------------------------
; MAIN QUESTION: PROBLEM OCCURRENCE STAGE
;-----------------------------------------------

(defrule ask-main-question
```

```
  ?f <- (ask-main-question)
  (not (problem_phase ?))
  =>
  (retract ?f)
  (printout t crlf "=== MAIN DIAGNOSTICS ===" crlf)
  (printout t "At what stage does the problem occur?" crlf)
  (printout t "1 - During boot" crlf)
  (printout t "2 - During system work" crlf)
  (printout t "Enter number (1-2): ")
  (bind ?answer (ask-with-validation "" 1 2))
  (if (= ?answer 1) then (assert (problem_phase boot)))
  (if (= ?answer 2) then (assert (problem_phase runtime)))
)

;-----------------------------------------------
; BRANCH 1: PROBLEMS DURING SYSTEM OPERATION
;-----------------------------------------------

(defrule ask-runtime-problem
  (problem_phase runtime)
  (not (runtime_problem_type ?))
  =>
  (printout t crlf "=== RUNTIME PROBLEMS ===" crlf)
  (printout t "What specific problem occurs?" crlf)
  (printout t "1 - Computer regulary shutdown" crlf)
  (printout t "2 - Operating system error" crlf)
  (printout t "3 - No internet/network" crlf)
  (printout t "4 - Performance degradation" crlf)
  (printout t "Enter number (1-4): ")
  (bind ?answer (ask-with-validation "" 1 2 3 4))
  (if (= ?answer 1) then (assert (runtime_problem_type shutdown)))
  (if (= ?answer 2) then (assert (runtime_problem_type os_error)))
  (if   (=   ?answer   3)   then   (assert   (runtime_problem_type
no_network)))
  (if   (=   ?answer   4)   then   (assert   (runtime_problem_type
performance)))
)

;--- 1.1 COMPUTER SHUTDOWN ---

(defrule ask-shutdown-type
  (runtime_problem_type shutdown)
  (not (shutdown_type ?))
  =>
  (printout t crlf "=== COMPUTER SHUTDOWN ===" crlf)
  (printout t "When does the shutdown occur?" crlf)
```

```
  (printout t "1 - Under load (games, heavy programs)" crlf)
  (printout t "2 - Randomly, without apparent reason" crlf)
  (printout t "Enter number (1-2): ")
  (bind ?answer (ask-with-validation "" 1 2))
  (if (= ?answer 1) then (assert (shutdown_type under_load)))
  (if (= ?answer 2) then (assert (shutdown_type random)))
)

; Rule 1.1.1: Shutdown under load
(defrule solution-overheating
  (shutdown_type under_load)
  (not (solution ?))
  =>
  (assert (solution replace_thermal_paste))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Component overheating" crlf)
  (printout t "SOLUTION: Replace thermal paste on CPU and GPU and
clean cooling system from dust" crlf)
)

; Rule 1.1.2: Random shutdown
(defrule ask-software-changes
  (shutdown_type random)
  (not (software_changes ?))
  =>
  (printout t crlf "=== RANDOM SHUTDOWN ===" crlf)
  (printout t "Were there recent software changes?" crlf)
  (printout t "1 - Yes" crlf)
  (printout t "2 - No" crlf)
  (printout t "Enter number (1-2): ")
  (bind ?answer (ask-with-validation "" 1 2))
  (if (= ?answer 1) then (assert (software_changes yes)))
  (if (= ?answer 2) then (assert (software_changes no)))
)

; Rule 1.1.2.1: Software changes were made
(defrule solution-rollback-changes
  (software_changes yes)
  (not (solution ?))
  =>
  (assert (solution rollback_changes))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Software error" crlf)
  (printout t "SOLUTION: Roll back recent software changes" crlf)
  (printout t "INSTRUCTIONS:" crlf)
  (printout t "1. System restore through restore point" crlf)
```

```
  (printout t "2. Remove recently installed programs" crlf)
  (printout t "3. Roll back system updates" crlf)
)

; Rule 1.1.2.2: No software changes
(defrule solution-reinstall-os
  (software_changes no)
  (not (solution ?))
  =>
  (assert (solution reinstall_os))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Critical OS error" crlf)
  (printout t "SOLUTION: Reinstall operating system" crlf)
)

;--- 1.2 OPERATING SYSTEM ERROR ---

(defrule solution-study-error-code
  (runtime_problem_type os_error)
  (not (solution ?))
  =>
  (assert (solution study_error_code))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Operating system error" crlf)
  (printout t "SOLUTION: Explore error code for precise diagnosis"
crlf)
)

;--- 1.3 INTERNET/NETWORK PROBLEMS ---

(defrule ask-network-problem-type
  (runtime_problem_type no_network)
  (not (network_problem_type ?))
  =>
  (printout t crlf "=== NETWORK PROBLEMS ===" crlf)
  (printout t "What exactly doesn't work?" crlf)
  (printout t "1 - No internet" crlf)
  (printout t "2 - No local network" crlf)
  (printout t "Enter number (1-2): ")
  (bind ?answer (ask-with-validation "" 1 2))
  (if   (=   ?answer   1)   then   (assert   (network_problem_type
no_internet)))
  (if   (=   ?answer   2)   then   (assert   (network_problem_type
no_network)))
)
```

```
; Rule 1.3.1: No internet
(defrule solution-configure-connection
  (network_problem_type no_internet)
  (not (solution ?))
  =>
  (assert (solution configure_connection))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: No internet connection" crlf)
  (printout t "SOLUTION:" crlf)
  (printout t "1. Restart router/modem" crlf)
  (printout t "2. Check IP and DNS settings" crlf)
  (printout t "3. Update network drivers" crlf)
  (printout t "4. Check internet on other devices in local network"
crlf)
  (printout t "5. Contact your provider" crlf)
)

; Rule 1.3.2: No network
(defrule ask-network-equipment
  (network_problem_type no_network)
  (not (network_equipment ?))
  =>
  (printout t crlf "=== NETWORK EQUIPMENT CHECK ===" crlf)
  (printout t "Is network equipment working properly?" crlf)
  (printout t "1 - Yes" crlf)
  (printout t "2 - No" crlf)
  (printout t "Enter number (1-2): ")
  (bind ?answer (ask-with-validation "" 1 2))
  (if (= ?answer 1) then (assert (network_equipment yes)))
  (if (= ?answer 2) then (assert (network_equipment no)))
)

; Rule 1.3.2.1: Equipment is working
(defrule solution-replace-network-card
  (network_equipment yes)
  (not (solution ?))
  =>
  (assert (solution replace_network_card))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Network card malfunction" crlf)
  (printout t "SOLUTIONS:" crlf)
  (printout t "1. Built-in network card - replace motherboard"
crlf)
  (printout t "2. External network card - replace with new one"
crlf)
)
```

```
; Rule 1.3.2.2: Equipment is faulty
(defrule solution-repair-network-equipment
  (network_equipment no)
  (not (solution ?))
  =>
  (assert (solution repair_network_equipment))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Network equipment problems" crlf)
  (printout t "SOLUTION: Repair/configure network equipment" crlf)
)

;--- 1.4 PERFORMANCE DEGRADATION ---

(defrule ask-performance-manifestation
  (runtime_problem_type performance)
  (not (performance_manifestation ?))
  =>
  (printout t crlf "=== PERFORMANCE DEGRADATION ===" crlf)
  (printout t "How does the problem manifest?" crlf)
  (printout t "1 - Slow OS or applications loading" crlf)
  (printout t "2 - System or applications performance degradation"
crlf)
  (printout t "Enter number (1-2): ")
  (bind ?answer (ask-with-validation "" 1 2))
  (if (= ?answer 1) then (assert (performance_manifestation
slow_loading)))
  (if (= ?answer 2) then (assert (performance_manifestation
performance_degradation)))
)

; Rule 1.4.1: Slow loading
(defrule solution-hdd-problem
  (performance_manifestation slow_loading)
  (not (solution ?))
  =>
  (assert (solution hdd_problem))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Hard drive issue" crlf)
  (printout t "SOLUTION:" crlf)
  (printout t "1. Check hard drive health" crlf)
  (printout t "2. Defragment disk (for HDD)" crlf)
  (printout t "3. Consider upgrading to SSD" crlf)
  (printout t "4. Free up disk space" crlf)
)
```

```
; Rule 1.4.2: Performance degradation
(defrule solution-overheating-or-virus
  (performance_manifestation performance_degradation)
  (not (solution ?))
  =>
  (assert (solution overheating_or_virus))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Overheating or virus infection" crlf)
  (printout t "SOLUTION:" crlf)
  (printout t "1. Check CPU and GPU temperatures" crlf)
  (printout t "2. Clean cooling system from dust" crlf)
  (printout t "3. Run antivirus scan" crlf)
  (printout t "4. Check for malware and unwanted programs" crlf)
  (printout t "5. Update drivers and operating system" crlf)
)

;------------------------------------------------
; BRANCH 2: PROBLEMS DURING BOOT
;------------------------------------------------

(defrule ask-boot-problem
  (problem_phase boot)
  (not (boot_problem_type ?))
  =>
  (printout t crlf "=== BOOT PROBLEMS ===" crlf)
  (printout t "What specific boot problem occurs?" crlf)
  (printout t "1 - Won't turn on" crlf)
  (printout t "2 - OS doesn't start loading" crlf)
  (printout t "3 - BIOS error" crlf)
  (printout t "Enter number (1-3): ")
  (bind ?answer (ask-with-validation "" 1 2 3))
  (if (= ?answer 1) then (assert (boot_problem_type
not_starting)))
  (if (= ?answer 2) then (assert (boot_problem_type
os_not_loading)))
  (if (= ?answer 3) then (assert (boot_problem_type bios_error)))
)

;--- 2.1 COMPUTER WON'T TURN ON ---

(defrule ask-fans-spinning
  (boot_problem_type not_starting)
  (not (fans_spinning ?))
  =>
  (printout t crlf "=== COMPUTER WON'T TURN ON ===" crlf)
  (printout t "Do fans spin when turning on?" crlf)
```

```
  (printout t "1 - Yes" crlf)
  (printout t "2 - No" crlf)
  (printout t "Enter number (1-2): ")
  (bind ?answer (ask-with-validation "" 1 2))
  (if (= ?answer 1) then (assert (fans_spinning yes)))
  (if (= ?answer 2) then (assert (fans_spinning no)))
)

; Rule 2.1.1: Fans are spinning
(defrule solution-replace-motherboard
  (fans_spinning yes)
  (not (solution ?))
  =>
  (assert (solution replace_motherboard))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Motherboard malfunction" crlf)
  (printout t "SOLUTION: Replace motherboard" crlf)
)

; Rule 2.1.2: Fans are not spinning
(defrule solution-replace-psu
  (fans_spinning no)
  (not (solution ?))
  =>
  (assert (solution replace_psu))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Power supply or cable malfunction" crlf)
  (printout t "SOLUTION: Replace power supply/power cable" crlf)
)

;--- 2.2 OS DOESN'T START LOADING ---

(defrule solution-reinstall-os-boot
  (boot_problem_type os_not_loading)
  (not (solution ?))
  =>
  (assert (solution reinstall_os_boot))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Critical OS damage" crlf)
  (printout t "SOLUTION: Reinstall operating system" crlf)
)

;--- 2.3 BIOS ERROR ---

(defrule ask-bios-beeps
  (boot_problem_type bios_error)
```

```
  (not (bios_beeps ?))
  =>
  (printout t crlf "=== BIOS ERROR ===" crlf)
  (printout t "What is the error content?" crlf)
  (printout t "1 - One short (or series of short) BIOS beeps" crlf)
  (printout t "2 - Long repeating BIOS beeps" crlf)
  (printout t "3 - One long two short BIOS beeps" crlf)
  (printout t "4 - \"Boot device not found\" error" crlf)
  (printout t "Enter number (1-4): ")
  (bind ?answer (ask-with-validation "" 1 2 3 4))
  (if (= ?answer 1) then (assert (bios_beeps ram_error)))
  (if (= ?answer 2) then (assert (bios_beeps cpu_error)))
  (if (= ?answer 3) then (assert (bios_beeps gpu_error)))
  (if (= ?answer 4) then (assert (bios_beeps no_boot_device)))
)

; Rule 2.3.1: RAM error
(defrule solution-replace-ram
  (bios_beeps ram_error)
  (not (solution ?))
  =>
  (assert (solution replace_ram))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: RAM malfunction" crlf)
  (printout t "SOLUTION: Replace RAM" crlf)
  (printout t "TESTING:" crlf)
  (printout t "1. Check each memory stick separately" crlf)
  (printout t "2. Try different memory slots" crlf)
  (printout t "3. Clean memory contacts" crlf)
)

; Rule 2.3.2: CPU error
(defrule solution-replace-cpu
  (bios_beeps cpu_error)
  (not (solution ?))
  =>
  (assert (solution replace_cpu))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: CPU malfunction" crlf)
  (printout t "SOLUTION: Replace CPU" crlf)
)

; Rule 2.3.3: GPU error
(defrule solution-replace-gpu
  (bios_beeps gpu_error)
  (not (solution ?))
```

```
  =>
  (assert (solution replace_gpu))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: GPU malfunction" crlf)
  (printout t "SOLUTION: Replace GPU" crlf)
  (printout t "OPTIONS:" crlf)
  (printout t "1. Use integrated graphics for testing" crlf)
  (printout t "2. Clean GPU contacts" crlf)
)

; Rule 2.3.4: No boot device
(defrule solution-replace-hdd
  (bios_beeps no_boot_device)
  (not (solution ?))
  =>
  (assert (solution replace_hdd))
  (printout t crlf "*** DIAGNOSIS COMPLETED ***" crlf)
  (printout t "PROBLEM: Hard drive malfunction" crlf)
  (printout t "SOLUTION: Replace hard drive" crlf)
)

;-----------------------------------------------
; SYSTEM COMPLETION
;-----------------------------------------------

(defrule system-completed
  (solution ?sol)
  =>
  (printout  t  crlf  "========================================="
crlf)
  (printout t "DIAGNOSTIC SYSTEM COMPLETED WORK" crlf)
  (printout t "=========================================" crlf)
  (halt)
)
```