

Лабораторная работа №1: Изучение основных возможностей и базовых команд среды CLIPS

Цель работы

Изучение основных возможностей и базовых команд среды продукционного программирования CLIPS и освоение способов разработки экспертной системы.

Основные теоретические положения

Среда CLIPS (C Language Integrated Production System) предназначена для построения экспертных систем (ЭС). Она поддерживает три основных способа представления знаний:

- продукционные правила для представления эвристических, основанных на опыте знаний;
- функции для представления процедурных знаний;
- объектно-ориентированное программирование.

Среда загружается запуском файла *clipswin.exe*. Назначение основных пунктов меню оконного интерфейса (версия 6.3), используемых при выполнении данного цикла лабораторных работ представлено в таблице ниже.

Пункт	Подпункт	«Горячие» клавиши	Назначение команды
File	New	^N	Вызов редактора
	Load	^L	Загрузка конструкций из файла
	Load Batch	-	Исполнение пакетного файла
Execution	Reset	^E	Инициализация конструкций
	Run	^R	Запуск МЛВ
	Step	^T	Выполнение одного шага вывода
Window	Facts Window	-	Активация окна списка фактов
	Agenda Window	-	Активация окна агенды

Для сброса среды CLIPS в исходное состояние используется команда (*clear*) или соответствующий пункт меню Execution.

Представление базовых типов данных. В CLIPS поддерживаются восемь простейших типов данных: целые числа (*integer*), числа с плавающей запятой (*float*), символьный (*symbol*), строковый (*string*), внешний адрес (*external-address*), адрес факта (*fact-address*), имя экземпляра (*instance-name*) и адрес экземпляра (*instance-address*).

Примеры записи числовых типов приведены ниже:

- Целое: 237, 15, +12, -32.
- С плавающей запятой: 237e3, 15.09, +12.0, -32.3e-7.

Символьный тип в CLIPS – любая последовательность символов, начинающаяся с отображаемого ASCII-символа и продолжающаяся до ограничителя. Ограничителем является любой неотображаемый ASCII-символ (пробел, табуляция, возврат каретки, перевод строки), кавычка, открывающая и закрывающая скобки, амперсанд (&), вертикальная черта (|), знак «меньше» (<) и тильда (~).

Строковый тип – множество отображаемых символов, заключенных в кавычки. Например: “abcd”, “fgs_85”, “foo#”, “13485*a”.

Другие типы в данной работе не используются.

Представление фактов и работа с ними. Факты являются одной из основных форм представления информации в CLIPS-системах и используются правилами для вывода новых фактов из имеющихся. Все текущие факты в CLIPS помещаются в список фактов (`fact-list`).

По формату представления в CLIPS выделяют два типа фактов: *упорядоченные* и *неупорядоченные*. В данной работе рассматриваются только упорядоченные факты. **Упорядоченный факт** состоит из заключенной в скобки последовательности одного или более полей, разделенных пробелами. Поля в неупорядоченном факте могут быть любыми простейшими типами данных (за исключением первого поля, которое должно быть символьного типа). Первое поле упорядоченного факта специфицирует отношение, которое применяется к остальным полям факта. Например:

- (высота 100);
- (включен насос);
- (студент Сидоров_Сергей);
- (однокурсники Иванов Петров Сидоров);
- (отец Иван Петр).

В последнем примере отношение является, некоммутативным, поэтому необходимо определить порядок аргументов, например «Иван является отцом Петра».

Для работы с фактами используются следующие команды: `assert` – добавляет факт в факт-список; `retract` – удаляет факт из списка; `modify` – модифицирует список; `duplicate` – дублирует факт. Например команда

```
(assert (length 150) (width 15) (weight "big"))
```

добавляет в список фактов три факта, каждый из которых состоит из двух полей.

Эти команды могут использоваться в режиме взаимодействия с пользователем или при выполнении CLIPS-программы. Некоторые команды, такие как `retract`, `modify` и `duplicate`, требуют, чтобы факты были идентифицированы. Для этой цели используется либо индекс факта (`fact-index`), либо адрес факта (`fact-address`). Индекс факта – уникальный целочисленный индекс, приписываемый факту всякий раз, когда факт добавляется (или модифицируется). Индекс факты начинаются с нуля и инкрементируются при каждом новом или измененном факте. Идентификатор факта (`fact identifier`) представляет собой краткую нотацию для отображения факта. Он состоит из символа “f”, за которым через тире следует индекс факта. Например, `f-10` ссылается на факт с индексом 10.

Для задания исходного множества фактов используется конструкция `deffacts`, со следующим

СИНТАКСИСОМ:

```
(deffacts <имя_группы_фактов> ["<комментарий>"] <факт>*)
```

где <имя_группы_фактов> - идентификатор символьного типа; <комментарий> - необязательное поле комментария; <факт>* - произвольная последовательность фактов, записанных через разделитель.

Пример использования конструкции deffacts:

```
(deffacts startup "Refrigerator Status"  
  (refrigerator light on)  
  (refrigerator door open))
```

Факты, определенные конструкцией deffacts добавляются в список фактов всякий раз при выполнении команды reset.

Для задания правил используется конструкция defrule со следующим синтаксисом:

```
(defrule <имя_правила> ["<комментарий>"]  
  [<объявление>]  
  <условный элемент>* ; Левая часть правила (антецедент)  
=>  
  <действие>* ; Правая часть правила (консеквент)
```

где <имя_правила> - идентификатор символьного типа, уникальный для данной группы правил; <комментарий> - необязательное поле комментария; <условный элемент>* - произвольная последовательность условных элементов; <действие>* - произвольная последовательность действий.

Пример задания правила:

```
(defrule R1 "Пример задания правила"  
  (days 2)  
  (works 100)  
=>  
  (printout t crlf "Свободного времени нет" crlf)  
  (assert (freetime "no")))
```

Данное правило содержит в левой части два условных элемента (упорядоченных факта), а в правой - команду printout вывода сообщения и команду assert добавления нового факта. В команде printout: t - параметр определяющий стандартный режим вывода, а - crlf символ возврата курсора и перевода его на новую строку.

Постановка задачи

Изучение базовых команд и конструкций CLIPS осуществляется посредством использования в среде CLIPS команд clear, reset, deffacts, defrule с пошаговой активизацией правил. Демонстрационная ЭС разрабатывается для предметной области, согласованной с

преподавателем и производится ее тестирование на различных комбинациях входных значений в пошаговом режиме.

Порядок выполнения работы

1. Изучение базовых команд и конструкций CLIPS

1. Запустить систему CLIPS (файл *clipswin.exe*). Активизировать окно просмотра текущего списка фактов (подпункт «Facts Window» пункта «Windows» главного меню). Выполнить следующую последовательность действий, фиксируя после каждого шага состояние списка фактов:
 - сбросить систему в исходное состояние командой (*clear*);
 - выполнить начальную установку командой (*reset*) или комбинацией клавиш $\wedge E$;
 - ввести 3 любых упорядоченных факта командой (*assert*), например:
`CLIPS> (assert (n n) (m m) (p p));`
 - повторно выполнить сброс командой (*reset*);
 - установить 3 ранее вводимых упорядоченных факта в качестве исходных фактов, используя конструкцию (*deffacts*);
 - выполнить сброс командой (*reset*).
2. Активизировать дополнительно окно просмотра агенды (подпункт «Agenda Window» пункта «Windows» главного меню). Выполнить следующую последовательность действий, фиксируя после каждого шага состояния списка фактов и агенды:
 - используя конструкцию (*defrule*), ввести три правила, такие, что антецеденты первых двух правил сопоставляются с комбинацией фактов, заданных ранее конструкцией (*deffacts*), а консеквенты этих правил добавляют новые факты, сопоставляемые с антецедентом третьего правила. Пусть, например, X, Y и Z – факты, заданные конструкцией (*deffacts*). Тогда структура вводимых правил может быть представлена следующим образом:

```
X & Y => V;  
Y & Z => W;  
V & W => U;
```
 - выполнить по шагам активизацию правил (используя «горячую» комбинацию $\wedge T$).

2. Разработка демонстрационной экспертной системы

- Сформировать, пользуясь редактором CLIPS, базу знаний демонстрационной ЭС и сохранить ее в файле *rulebase.CLP*. Предметную область экспертной системы выбрать по согласованию с преподавателем.
- Общее количество правил в базе знаний (БЗ) должно быть не менее 25. Количество значений переменных должно выбираться таким образом, чтобы БЗ отвечала требованию полноты, т.е. содержала правила, соответствующие любым сочетаниям значений переменных в левых частях правил. Например, если переменная «свободное время» имеет 3 значения («отсутствует», «мало» и «много»), а переменная «погода» – 2 значения («плохая» и «хорошая»), то максимальное число правил для определения переменной «действие» будет равно 6.

- В качестве примера, использовать фрагмент базы знаний, содержащийся в файле *rulebase.CLP*.
- Для активизации ЭС в среде CLIPS использовать пакетный файл *run_lab1.BAT*, который может быть запущен с использованием пункта «Load Batch» меню «File».
- Оттестировать ЭС на различных комбинациях входных значений в пошаговом режиме. Продемонстрировать работу ЭС преподавателю.

Содержание отчёта

- Цель работы.
- Краткое изложение основных теоретических понятий.
- Постановка задачи с кратким описанием порядка выполнения работы.
- Демонстрация работы базовых команд CLIPS.
- Структура ЭС.
- Демонстрация работы ЭС.
- Общий вывод по проделанной работе.
- Код программы.

Пример выполнения задания

Данная ЭС вырабатывает рекомендации студенту накануне зачета и имеет четыре входные переменные («число дней до зачета», «количество несделанных лабораторных работ (в %)», «температура на улице» и «наличие осадков»), две промежуточные («свободное время» и «погода») и выходную переменную («рекомендуемые действия»). Диаграмма зависимости переменных показана на рис. 1, в скобках указаны возможные имена переменных.



Реализация ЭС в среде CLIPS

[rulebase.CLP](#)

```
(defrule data-input
  (initial-fact)
=>
  (printout t crlf "Введите число дней до зачета (целое значение): ")
  (bind ?days (read))

  (if (numberp ?days)
    then (assert (days ?days))
    else (printout t "Введите число" crlf))

  (printout t crlf "Введите число несделанных лабораторных работ (в %): ")
  (bind ?works (read))
  (assert (works ?works)))

;;;;;;;;;;;;;
```

;;;;;;;;;

```
(defrule R1
  (days ?days)
  (works ?works)
  (test (and (= ?days 1) (<> ?works 0)))
=>
  (printout t crlf crlf "Свободного времени нет!" crlf)
  (assert (freetime "no")))
```

```
(defrule R2
  (days ?days)
  (works ?works)
  (test (and (= ?days 2) (>= ?works 10)))
=>
  (printout t crlf crlf "Свободного времени нет!" crlf)
  (assert (freetime "no")))
```

```
(defrule R3
  (days ?days)
  (works ?works)
  (test (and (= ?days 2) (< ?works 10)))
=>
  (printout t crlf crlf "Свободного времени мало!" crlf)
  (assert (freetime "a-little")))
```

```
(defrule R4
  (days ?days)
  (works ?works)
  (test (and (= ?days 3) (> ?works 25)))
=>
  (printout t crlf crlf "Свободного времени нет!" crlf)
  (assert (freetime "no")))
```

```
;RULE:   R5
;        IF:      days = 3 AND works <= 25 AND works > 10
;        THEN:    fretim = "little"
;RULE:   R6
;        IF:      days = 3 AND works <= 10
;        THEN:    fretim = "many"
;RULE:   R7
;        IF:      days = 4 AND works < 25
;        THEN:    fretim = "many"
;RULE:   R8
;        IF:      days = 4 AND works >= 25 AND works < 75
;        THEN:    fretim = "little"
;RULE:   R9
;        IF:      days = 4 AND works >= 75
;        THEN:    fretim = "no"
;RULE:   R10
```

```
;      IF:      days = 5 AND works < 60
;      THEN:    fretim = "many"
;RULE:  R11
;      IF:      days = 5 AND works >= 60 AND works < 90
;      THEN:    fretim = "little"
;RULE:  R12
;      IF:      days = 5 AND works >= 90
;      THEN:    fretim = "no"
;RULE:  R13
;      IF:      days > 5
;      THEN:    fretim = "many"
```

Пакетный файл

[run_lab1.BAT](#)

```
; Данный файл загружает базу знаний,
; инициализирует среду и запускает ЭС
(load rulebase.CLP)
(reset)
(run)
```

From:
<http://se.moevm.info/> - **se.moevm.info**

Permanent link:
http://se.moevm.info/doku.php/courses:knowledge_representation_and_artificial_intelligence_systems:lab1?rev=1569777255

Last update: **2022/12/10 09:08**