ДЕФИНИРАНЕ НА ФУНКЦИИ В CLIPS

```
(deffunction <name> [<comment>]
     (<regular-parameter>*
     [<wildcard-parameter>])
     <action>*)
```

Пример

```
> (clear)
> (deffunction print-args
    (?a ?b $?c)
    (printout t ?a " " ?b " and "
    (length ?c) " extras: " ?c
    crlf))
> (print-args 1 2)
1 2 and 0 extras: ()
> (print-args a b c d)
a b and 2 extras: (c d)
```

Пример за дефиниция на функция, която съдържа пряка рекурсия

```
(deffunction factorial (?n)
   (if (or (not (integerp ?n))
           (< ?n 0))
       then (printout t "Factorial
 error!" crlf)
       else (if (= ?n 0) then 1 else
 (* ?n (factorial (- ?n 1))))))
```

Косвена рекурсия

Forward декларация: включва само заглавната част на дефиницията на функцията (без тялото ѝ).

```
(deffunction f2 ())
(deffunction f1 () (f2))
(deffunction f2 () (f1))
```

Генерични (родови) функции

```
(defgeneric <name> [<comment>])
(defmethod <name> [<index>] [<comment>]
   (<parameter-restriction>*
     [<wildcard-parameter-restriction>])
   <action>*)
<parameter-restriction> ::=
   <single-field-variable> |
   (<single-field-variable> <type>* [<query>])
<wildcard-parameter-restriction> ::=
   <multifield-variable> |
   (<multifield-variable> <type>* [<query>])
```

Допълнително (quiery) ограничение: булев тест, дефиниран от потребителя, който трябва да бъде удовлетворен от съответния фактически параметър.

```
<query> ::= <global-variable> |
  <function-call>
```

Пример

```
> (clear)
> (defmethod +
     ((?a STRING) (?b STRING))
      (str-cat ?a ?b))
> (+ 1 2)
3
> (+ "ala" "bala")
"alabala"
>
```

Цитиране на текущото поле на незадължителния параметър: променлива **?current-argument**

Пример

```
> (clear)
> (defmethod +
    (($?any INTEGER
        (evenp ?current-argument)))
     (div (call-next-method) 2))
> (+ 1 2)
3
> (+ 4 6 4)
7
```

Правила за намиране на по-приоритетния от два приложими метода

1. По-специфичното ограничение по тип (върху типа) за даден параметър има приоритет.
2. Параметър с quiery ограничение има по-висок приоритет от такъв без quiery ограничение.
3. Задължителен параметър има приоритет пред незадължителен.
4. Методът с по-голям брой задължителни параметри е по-приоритетен.
5. Метод, който няма незадължителен параметър, е по-приоритетен от метод, който има незадължителен параметър.
6. Метод, който е дефиниран преди друг метод, има приоритет.

Пример 1

```
(defmethod f ((?a NUMBER STRING)))          ;;; #1
(defmethod f ((?a INTEGER LEXEME)))         ;;; #2
```

Ред: #2, #1 (INTEGER е подклас на NUMBER).

Пример 2

```
(defmethod f ((?a MULTIFIELD STRING)))      ;;; #1
(defmethod f ((?a LEXEME)))                 ;;; #2
```

Ред: #2, #1 (MULTIFIELD и LEXEME са несравними, #2 има по-малко елементи в списъка на класовете).

Пример 3

```
(defmethod f ((?a INTEGER LEXEME))     ;;; #1
(defmethod f ((?a STRING NUMBER))      ;;; #2
```

Ред: #1, #2 (съответните двойки от класове са несравними, определящ е редът на дефиниране на двата метода).

Забележка. Класът LEXEME е дефиниран като обединение на SYMBOL и STRING.

Конструкции за работа с генерични функции

`(get-defgeneric-list)`

Връща списък от имената на всички генерични функции, които са дефинирани до момента.

`(get-defmethod-list [<name>])`

Връща списък от дефинициите на всички дефинирани методи (за дадена генерична функция).

(list-defmethods [<name>])

Връща списък от всички дефинирани методи (за дадена генерична функция).

(call-next-method)

Извиква следващия по специфичност метод, който е приложим към същите фактически параметри, които е получил текущо изпълняваният метод.

(override-next-method <expression>*)

Измежду методите, които са по-общи от текущо изпълнявания, се избира и изпълнява (върху новите фактически параметри) най-специфичният метод, който е приложим към новите фактически параметри.

Пример

```
> (clear)
> (defmethod +
    ((?a INTEGER) (?b INTEGER))
     (override-next-method
         (* ?a 2) (* ?b 3)))
> (list-defmethods +)
+ #2 (INTEGER) (INTEGER)
+ #SYS1 (NUMBER) (NUMBER) ($? NUMBER)
> (+ 1 2)
8
```

```
(call-specific-method <name>
      <method-index> <expression>*)
```

Тази функция позволява да бъде изпълнен конкретен метод (който се задава чрез неговия индекс), без да има значение неговата специфичност. Посоченият метод трябва да бъде приложим към дадените фактически параметри.

Пример

```
> (clear)
> (defmethod +
     ((?a INTEGER) (?b INTEGER))
      (* (- ?a ?b)(- ?b ?a)))
> (+ 1 2)
-1
> (call specific-method + 1 1 2)
3
```