

Построение новых функций в среде Common Lisp

Вычисляемые функции

λ -выражение в исчислении Черча

Определение функций и их вычисление в языке **LISP** основано на *λ -исчислении Черча*.

λ –выражение является элементом

λ -исчисления и важным механизмом в практическом программировании.

В λ -исчислении Черча функция записывается в следующем виде:

$\lambda (x_1, x_2, \dots, x_N).f_N$.

В языке **LISP** λ -выражение имеет вид:

(LAMBDA (X1 X2 ... XN) FN)

λ -выражение в языке Лисп

В языке **LISP** λ -выражение имеет вид:

(LAMBDA (X1 X2 ... XN) FN)

Функция **LAMBDA** предназначена для определения "безымянных" (неименованных) функций и называется ***вычисляемой функцией*** (не следует путать с понятием вычислимой функции в теории алгоритмов!).

Первый аргумент вычисляемой функции - **(X1 X2 ... XN)** является списком (возможно, пустым!). Его называют **λ-списком**. **X1, X2, ..., XN** называются ***формальными параметрами***.

Второй аргумент функции **LAMBDA - FN** называется ***телом***. Оно представляет собой произвольное выражение, значение которого может вычислить интерпретатор языка **LISP**.

λ -выражение в языке Лисп

Лямбда-выражение



(LAMBDA (X1 X2 ... XN) FN)

Имя функции

Лямбда-список

Тело

λ -ВЫЗОВ

Для того, чтобы применить λ -функцию к аргументам, нужно в вызове функции поставить λ -выражение вместо имени функции:

((LAMBDA (X1 X2 ... XN) FN) A1 A2 ... AN)

Здесь **A_i ($i=1,2,\dots,N$)** - выражения языка **LISP**,
определяющие ***фактические параметры***.

Такую форму вызова называют **λ -вызовом**.

Вычисление λ -выражения

Вычисление λ -вызова (применение λ -выражения к фактическим параметрам) производится в два этапа. Сначала вычисляются значения фактических параметров и соответствующие формальные параметры связываются с полученными значениями. Этот этап называется **связыванием параметров**. На следующем этапе с учетом новых связей вычисляется тело λ -выражения, и полученное значение возвращается в качестве значения λ -вызова. Формальным параметрам после окончания вычисления возвращаются те связи, которые у них были перед вычислением λ -вызова. Весь этот процесс называют **λ -преобразованием (λ -конверсией)**.

Примеры λ -преобразований

\$ ((LAMBDA (NUM) (PLUS NUM 1)) 45)

46

\$ ((LAMBDA (M N) (COND ((LESSP M N) M) (T N))) 233 123)

233

\$ (LAMBDA NIL (PLUS 3 5))

8

\$ (LAMBDA () (PLUS 3 5))

8

\$ ((LAMBDA (X) (EQ X NIL)) NIL)

T

Особенности использования λ -преобразований

λ -выражение - это "безымянная" функция, которая пропадает тотчас же после λ -преобразования. Ее трудно использовать снова, так как нельзя вызвать по имени, хотя λ -вызов доступен как списочный объект.

"Безымянные" функции используются, например, при передаче функции в качестве аргумента другой функции или при формировании функции в результате вычислений, другими словами, при ***синтезе программ***.

Пример определения функции с помощью конструкции LAMBDA

Пусть требуется описать функцию $y=F(x)$ в зависимости от условия с помощью конструкции LAMBDA :

$$Y = \begin{cases} X^2, & \text{если } X \leq 2, \\ X + 5, & \text{если } 2 < X < 6, \\ X - 2, & \text{если } X \geq 6 \end{cases}$$

Пример1 определения функции с помощью конструкции LAMBDA

```
((LAMBDA (X)
  (COND
    ( (<= X 2) (* X X))
    ((AND (> X 2) (< X 6)) (+ X 5))
    (T (- X 2)))) 3)
```

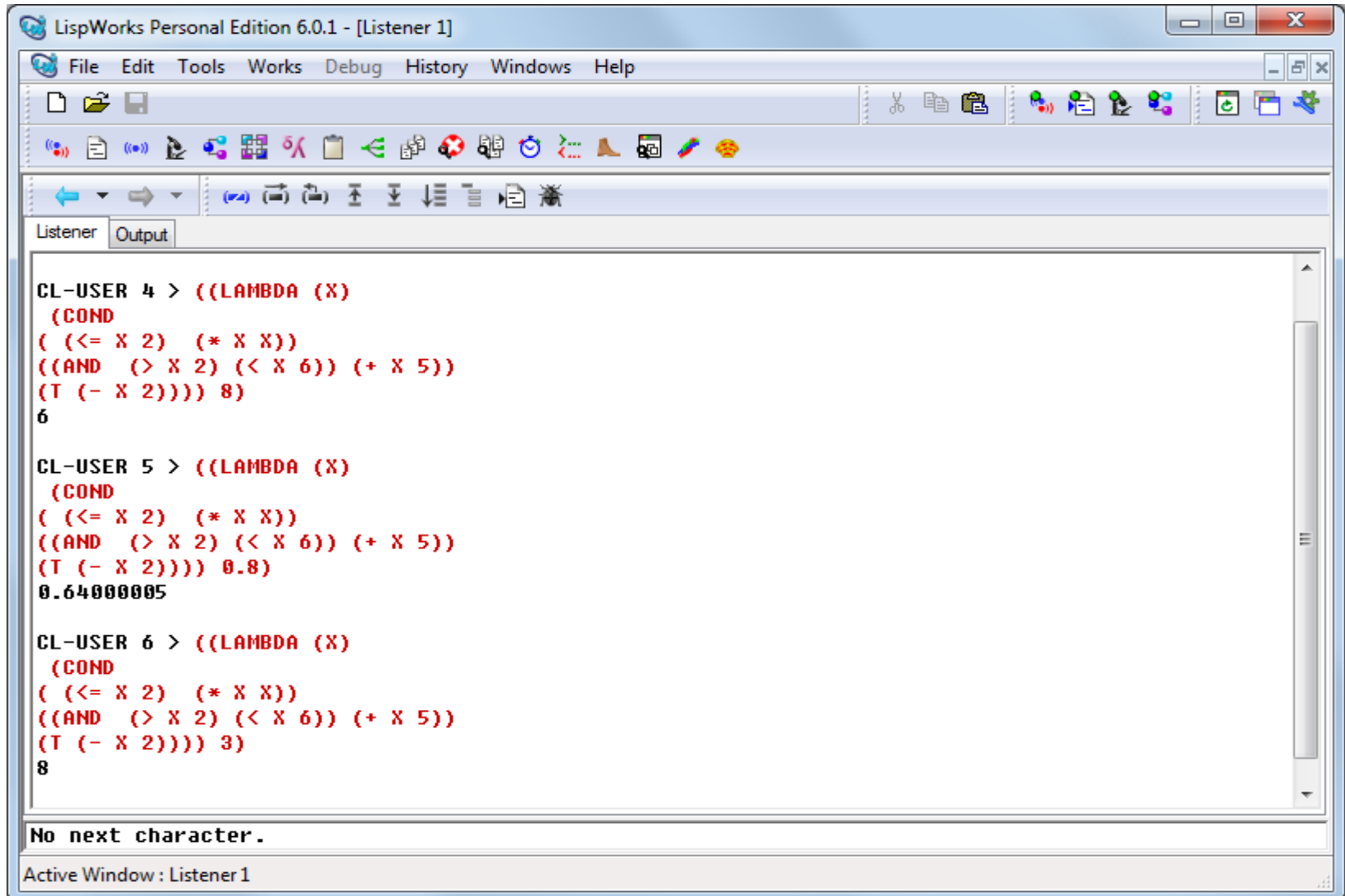
Пример2 определения функции с помощью конструкции LAMBDA

```
((LAMBDA (X)
  (COND
    ( (<= X 2) (* X X))
    ((AND (> X 2) (< X 6)) (+ X 5))
    (T (- X 2)))) 8)
```

Пример3 определения функции с помощью конструкции LAMBDA

```
((LAMBDA (X)
  (COND
    ( (<= X 2) (* X X))
    ((AND (> X 2) (< X 6)) (+ X 5))
    (T (- X 2)))) 0.8)
```

Примеры λ -преобразований



The screenshot shows the LispWorks Personal Edition 6.0.1 - [Listener 1] window. The interface includes a menu bar (File, Edit, Tools, Works, Debug, History, Windows, Help), a toolbar with various icons, and a main workspace divided into 'Listener' and 'Output' tabs. The 'Listener' tab is active, displaying three lambda expressions being evaluated. The first expression returns 6, the second returns 0.64000005, and the third returns 8. The status bar at the bottom indicates 'Active Window : Listener 1'.

```
CL-USER 4 > ((LAMBDA (X)
  (COND
    ( (<= X 2) (* X X))
    ((AND (> X 2) (< X 6)) (+ X 5))
    (T (- X 2)))) 8)
6

CL-USER 5 > ((LAMBDA (X)
  (COND
    ( (<= X 2) (* X X))
    ((AND (> X 2) (< X 6)) (+ X 5))
    (T (- X 2)))) 0.8)
0.64000005

CL-USER 6 > ((LAMBDA (X)
  (COND
    ( (<= X 2) (* X X))
    ((AND (> X 2) (< X 6)) (+ X 5))
    (T (- X 2)))) 3)
8

No next character.
```

Active Window : Listener 1

Построение новых функций в среде Common Lisp

**Именованные функции
(функция DEFUN)**

Функция DEFUN

Определить новую функцию и дать ей имя можно с помощью функции **DEFUN** (**DE**fine **FUN**ction).

Функция **DEFUN** вызывается так:

```
(DEFUN  
имя_функции(список_формальных_параметров)  
тело_функции  
)
```

Тело функции – это последовательность вызовов уже определенных функций.

Функция **DEFUN** возвращает имя новой функции.

После такого описания к функции можно обращаться в данном сеансе работы интерпретатора Лисп.

Формальные параметры функции

Формальные параметры функции называют еще *лексическими или статическими переменными*.

Связи статической переменной действительны только в пределах той функции, в которой они определены.

Они перестают действовать в функциях, вызываемых во время вычисления, но текстуально описанных вне данной функции.

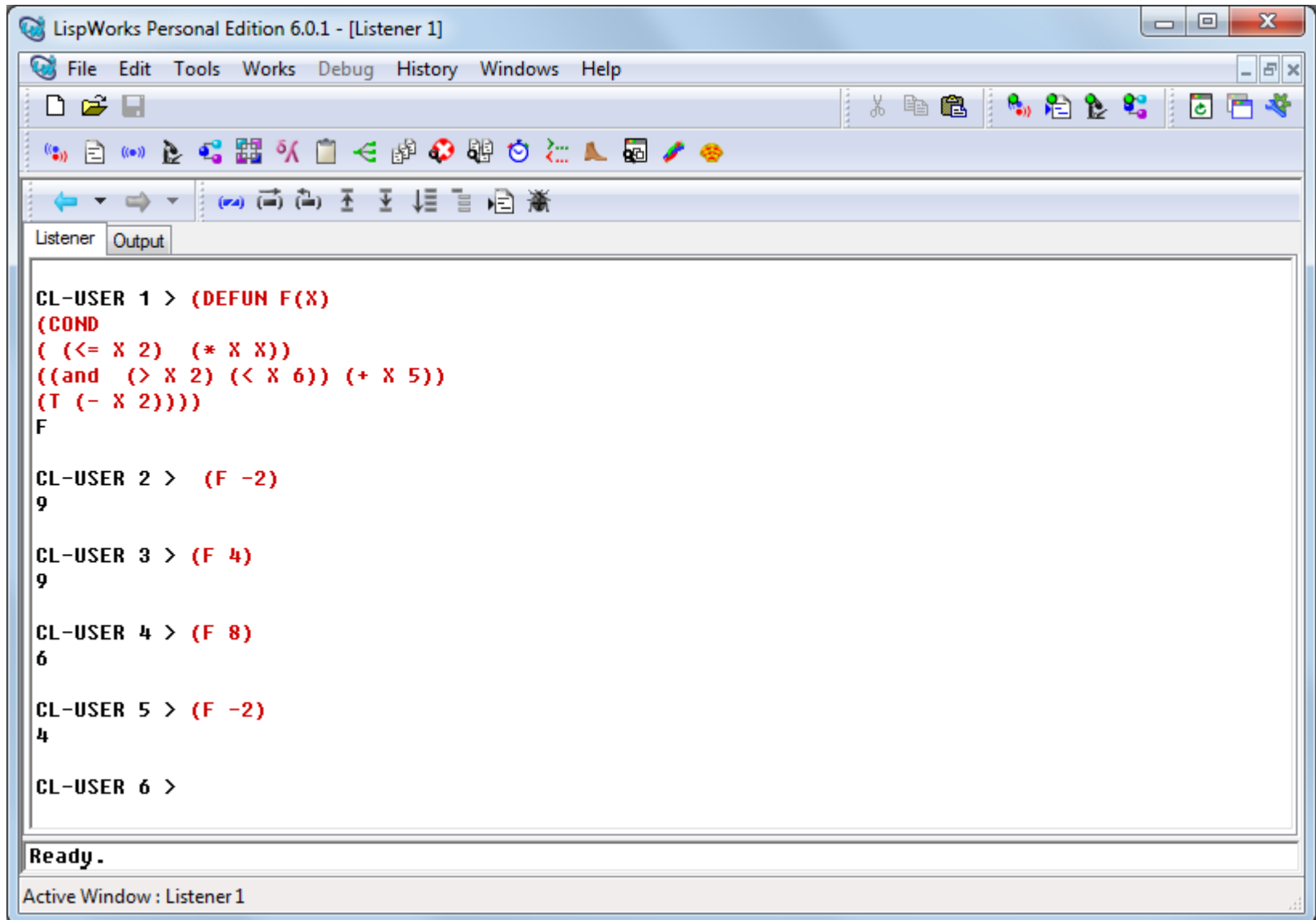
После вычисления функции, созданные на это время связи формальных параметров ликвидируются и происходит возврат к тому состоянию, которое было до вызова функции.

Пример определения функции с помощью конструкции DEFUN

Пусть требуется описать функцию $y=F(x)$ в зависимости от условия с помощью конструкции DEFUN:

$$Y = \begin{cases} X^2, & \text{если } X \leq 2, \\ X + 5, & \text{если } 2 < X < 6, \\ X - 2, & \text{если } X \geq 6 \end{cases}$$

Пример определения функции с помощью конструкции



The screenshot shows the LispWorks Personal Edition 6.0.1 window. The title bar reads "LispWorks Personal Edition 6.0.1 - [Listener 1]". The menu bar includes File, Edit, Tools, Works, Debug, History, Windows, and Help. Below the menu bar is a toolbar with various icons. The main window is divided into two panes: "Listener" and "Output". The "Listener" pane shows the following code:

```
CL-USER 1 > (DEFUN F(X)
(COND
( (<= X 2) (* X X))
((and (> X 2) (< X 6)) (+ X 5))
(T (- X 2))))
F
```

The "Output" pane shows the results of the function calls:

```
CL-USER 2 > (F -2)
9

CL-USER 3 > (F 4)
9

CL-USER 4 > (F 8)
6

CL-USER 5 > (F -2)
4

CL-USER 6 >
```

At the bottom of the window, the status bar displays "Ready ." and "Active Window : Listener 1".

Рекурсивные функции

Рекурсивная функция имеет следующую структуру:

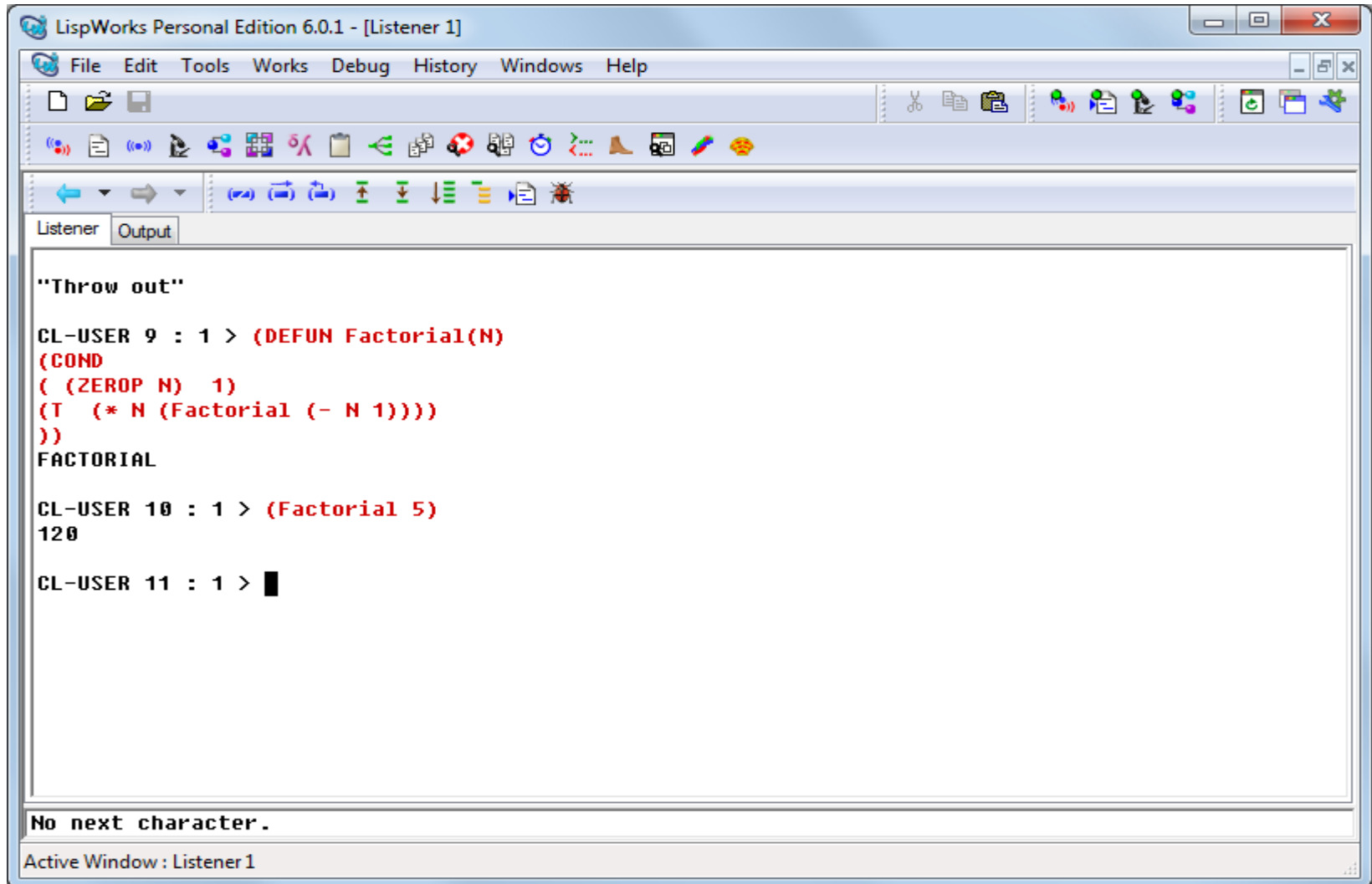
```
(DEFUN имя_функции(список_формальных_параметров)
(COND
(P1 S1)
(P2 S2)
.....
(Pn Sn)
))
```

где P_i – предикаты;

S_i – выражения произвольной формы.

Причем не менее одно S_i должно содержать имя определяемой функции.

Пример1 рекурсивной функции. Определение факториала.

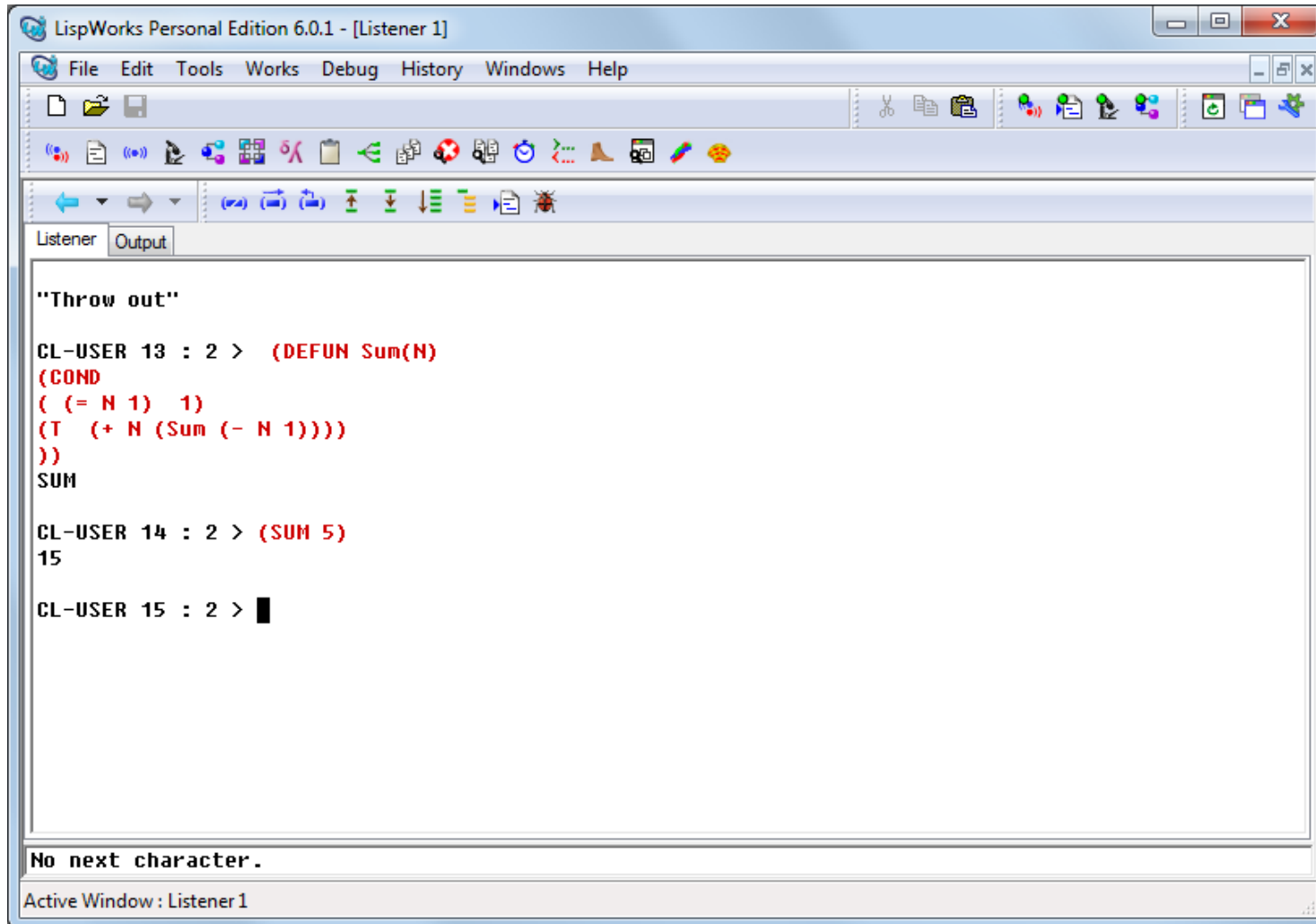


The screenshot shows the LispWorks Personal Edition 6.0.1 - [Listener 1] window. The interface includes a menu bar (File, Edit, Tools, Works, Debug, History, Windows, Help), a toolbar with various icons, and a main workspace with 'Listener' and 'Output' tabs. The 'Listener' tab is active, displaying the following text:

```
"Throw out"  
  
CL-USER 9 : 1 > (DEFUN Factorial(N)  
(COND  
( (ZEROP N) 1)  
(T (* N (Factorial (- N 1)))))  
)  
FACTORIAL  
  
CL-USER 10 : 1 > (Factorial 5)  
120  
  
CL-USER 11 : 1 > █
```

At the bottom of the window, the status bar displays "No next character." and "Active Window : Listener 1".

Пример2 рекурсивной функции. Определение суммы ряда натуральных чисел.



The screenshot shows the LispWorks Personal Edition 6.0.1 interface. The title bar reads "LispWorks Personal Edition 6.0.1 - [Listener 1]". The menu bar includes "File", "Edit", "Tools", "Works", "Debug", "History", "Windows", and "Help". Below the menu bar is a toolbar with various icons. The main window has two tabs: "Listener" and "Output". The "Listener" tab is active and displays the following text:

```
"Throw out"  
  
CL-USER 13 : 2 > (DEFUN Sum(N)  
(COND  
  ( (= N 1) 1)  
  (T (+ N (Sum (- N 1)))))  
)  
SUM  
  
CL-USER 14 : 2 > (SUM 5)  
15  
  
CL-USER 15 : 2 > █
```

At the bottom of the window, the status bar displays "No next character." and "Active Window : Listener 1".