



LISP

ПЕРЕМЕННЫЕ, ФУНКЦИИ И РЕКУРСИЯ

СТУДЕНТЫ ГРУППЫ 5030102/20202:
ГЕОРГИЙ ШТЕЙНБЕРГ
СЕРГЕЙ ТИШКОВЕЦ

ПЛАН ПРЕЗЕНТАЦИИ

- Объявление переменных
- Области видимости
- Определение функции
- Особенности функций
- Рекурсия
- Замыкания (Closures)
- Список литературы

ПЕРЕМЕННЫЕ: ОБЪЯВЛЕНИЕ И СВЯЗЫВАНИЕ

Переменные не объявляются с типом — **они связываются со значением**

- **Присваивание:**

- **(*setq variable value*)** — присваивает значение существующей переменной

Пример: **(*setq x 15*)**

- **Глобальные переменные:**

- ***defvar*** — определяет переменную, инициализируемую один раз. При повторной загрузке кода значение сохраняется

Пример: **(*defvar db_connection nil*)**

- ***defparameter*** - определяет параметр, значение которого можно переопределять

Пример: **(*defparameter debug_mode t*)**

ПЕРЕМЕННЫЕ: ОБЪЯВЛЕНИЕ И СВЯЗЫВАНИЕ

- **Локальные переменные:**

- ***let*** – связывает переменные параллельно. Выражения инициализации вычисляются до создания связей

```
12 (handler-case
13   | (let ((a 10)
14         | | (b a))                ; a ещё не связано, b будет неопределённым
15   | | (print b))                  ; Вывод: NIL
16   (error (e) (format t "Error in let: ~a~%" e)))
```

- ***let**** – связывает переменные последовательно. Каждое следующее выражение инициализации видит предыдущие переменные

```
18 (let* ((a 10)
19        | | | (b a))              ; a уже связано, b = 10
20        | (print b))              ; Вывод: 10
21
```

ОБЛАСТИ ВИДИМОСТИ

Лексическая (статическая) – по умолчанию для *let*

- Переменная видна в точке определения и во всех вложенных конструкциях

```
24  (let ((x 100))
25    (defun test-lexical ()
26      x))
27  (let ((x 200))                ; Попытка переопределить x
28    (print (test-lexical)))      ; Вывод: 100 (лексическое значение)
```

Динамическая (специальная) – для *defvar/defparameter*

- Переменная видна всем функциям, вызванным во время выполнения блока

```
31  (defvar *dynamic-var* 50)
32  (defun test-dynamic ()
33    *dynamic-var*)
34  (let ((*dynamic-var* 75))
35    (print (test-dynamic)))      ; Вывод: 75 (динамическое значение)
36  (print (test-dynamic))         ; Вывод: 50 (исходное значение)
```

ФУНКЦИИ В LISP

- Определение функции

```
39 (defun square (x)
40   |  (* x x))
41 (print (square 4))
```

- Анонимные функции: *lambda*

```
43 (print (funcall (lambda (x) (* x x)) 5))
```

ФУНКЦИИ В LISP: ОСОБЕННОСТИ

- Множественные возвращаемые значения:

```
46 (defun divide (a b)
47   (values (truncate a b) (mod a b)))
48 (multiple-value-bind (quotient remainder) (divide 10 3)
49   (print quotient)           ; Вывод: 3
50   (print remainder))        ; Вывод: 1
51
```

- Опциональные аргументы (*&optional*):

```
52 (defun greet (name &optional (greeting "Hello"))
53   (format nil "~a, ~a!" greeting name))
54 (print (greet "Alice"))      ; Вывод: "Hello, Alice!"
55 (print (greet "Bob" "Hi"))   ; Вывод: "Hi, Bob!"
```

РЕКУРСИЯ

- **Факториал**

```
58 (defun factorial (n)
59   (if (<= n 1)
60     1
61     (* n (factorial (- n 1)))))
62 (print (factorial 5))           ; Вывод: 120
```

- **Обработка списков**

```
65 (defun my-list-length (lst)
66   (if (null lst)
67     0
68     (+ 1 (my-list-length (cdr lst)))))
69 (print (my-list-length '(1 2 3 4))) ; Вывод: 4
```

- **Хвостовая рекурсия** – когда рекурсивный вызов является последней операцией

```
71 √ (defun factorial-tail (n &optional (acc 1))
72 √   (if (<= n 1)
73       acc
74       (factorial-tail (- n 1) (* n acc))))
75 (print (factorial-tail 5))           ; Вывод: 120
```


ЗАМЫКАНИЯ (CLOSURES)

- Функции, которые "запоминают" лексическое окружение своего создания

```
78 (defun make-counter ()  
79   (let ((count 0))  
80     (lambda () (incf count))))  
81 (defvar counter1 (make-counter))  
82 (defvar counter2 (make-counter))  
83 (print (funcall counter1))      ; Вывод: 1  
84 (print (funcall counter1))      ; Вывод: 2  
85 (print (funcall counter2))      ; Вывод: 1  
86
```

Применение:

- Создание функций с состоянием.
- Реализация объектов и инкапсуляции данных.
- Callback-функции, которым нужно запоминать контекст



СПИСОК ЛИТЕРАТУРЫ

- [LISP - Краткое руководство](#)
- [Основы программирования на языке Lisp](#)

Abstract geometric lines in white on a black background, forming various polygons and intersecting lines on the left side of the slide.

СПАСИБО ЗА ВНИМАНИЕ