UCLA Samueli Computer Science

Lisp Programming Fundamentals

Winter 2023 CS 35L LAs: Paul Serafimescu & Kosta Gjorgjievski

UCLA Samueli Computer Science

Program Structure & Basic Syntax

- Created at MIT in 1958 by John McCarthy
  - It's *really* old (for reference C was 1970s)
  - Version we use is Emacs Lisp, which is one dialect
  - Other dialects include Scheme, Racket, Common Lisp which are more or less the same
- "Polish" notation
  - `(operation operator_1 operator_2)`
  - example: `(+ 2 1)` results in `3`
- Program is defined by top-level functions
  - Top-level means at the "global" level

UCLA Samueli Computer Science

Data Types

- `integer`, `float`, `string`
  - usual data types
- `symbol`
  - usually created by quoting (we will cover this later in depth)
- `cons`
  - the basic list structure (will cover later)
  - linked list structure “under the hood”
- `t` & `nil`
  - “boolean” values
  - `nil` is like `NULL` in C or `nullptr` in C++ as well

Variables

- These don't really exist in Lisp.
- However, there are macros which set global variables, and bindings locally
- `(setq variable_name value)`
  - Sets a global variable `variable_name` to value `value`
  - This usually isn't good practice
  - Emacs has an interactive command `set-variable` which can be run through M-x
  - What is the difference between interactive and non-interactive in ELisp?
- `let` bindings (local variables)

UCLA Samueli Computer Science

Control Flow

- Two possibilities
- `(cond (cond_1 action_1) (cond_2 action_2)...(t action_n))`
  - similar(ish) to a switch
  - check each condition and execute the first matched action, default is `t` (true is always executed)
- `(if condition action_1 action_2)`
  - if condition is truthy (not `nil`) execute `action_1` otherwise execute `action_2`
- Matter of preference which you want to use
  - I prefer `cond`

UCLA Samueli Computer Science

Lists and Cons

- Cons is another name for the list in Lisp, but it's also the function that creates a new "node" of your linked list
  - always created in the front (the head)
  - the end of the list is delimited by `nil`
- `(cons 1 (cons 2 (cons 3 nil)))` creates a list `(1, 2, 3)`
  - think of it like a singly linked list
- Quoting is another way to create lists quickly (usually static)
  - `'(1 2 3)` creates a list `(1, 2, 3)`

UCLA Samueli Computer Science

Lists and Cons (continued)

- Quoting is a way to create symbols without evaluating them
  - Can also be done by a `quote` function
    - `'(1 2 3)` is equivalent to `(quote (1 2 3))`
  - `'x` creates a symbol `x`, it is not looking at the value of some variable `x`
    - this is pretty trippy and probably won't be tested on but it's still cool
    - based on what you know about list creation, what does `'()` evaluate to?
- Quick exercise: what does `'(cons 1 (cons 2 nil))` resolve to?
- If you think about it, a Lisp program is just one big quoted Lisp list
  - What benefits does this have?

UCLA Samueli Computer Science

Recursion

- You probably noticed we did not cover any loop syntax
  - There is none, we have to use recursion and pattern matching
  - Functional programming (covered in depth in CS 131 which Prof. Eggert teaches too)
- You can use `car` and `cdr` + conditionals + recursion to iterate over a list.
  - `car` returns the element at the front of the list
  - `cdr` returns the rest of the list
    - other functions like `(cadr x)`, which is equivalent to `(car (cdr x))`
- How would you check if an element is in a list of integers?

UCLA Samueli Computer Science

Have a good weekend!

We hope you learned something too