(((((((Lisp)))))))

Adam Blažek

12. listopadu 2024

18OOP, FJFI ČVUT

Lisp

```
(je (LISP)
  (rodina)
  (jazyků
    (programovacích)
    :s (historií (dlouhou))
    :které (přinesly
      (spoustu (inovace))
      :do (návrhu (jazyků (programovacích))))))
```

Lisp

Lisp je rodina programovacích jazyků s dlouhou historií, které přinesly spoustu inovace do návrhu programovacích jazyků.

Počátky

Lisp = **Lis**t **p**rocessor.

Vymyslel ho John McCarthy v roce 1980.

V roce 1960 publikován článek *Recursive Functions of Symbolic Expressions and Their Computation by Machine*.

Jazyk byl zamýšlen jako čistě teoretický, ale Steve Russell naprogramoval interpreter pro počítač IBM 704.

V sedmdesátých letech byl využíván pro umělou inteligenci.

Prvenství

Lisp je (téměř) první jazyk, který přinesl:

- Dynamické typy
- Rekurzi
- Automatickou správu paměti (*garbage collection*)
- Funkce vyššího řádu
- Stromovité datové struktury
- Upravování kódu za běhu
- Interaktivitu (REPL = **R**ead-**E**valuate-**P**rint **L**oop)
- Symboly jako datový typ
- Nerozlišování mezi příkazy a výrazy

Něco z toho už dříve v IPL (**I**nformation **P**rocessing **L**anguage)

S-výrazy

S-výrazy

```
(defun factorial (n)
  (if (= n 0)
    1
    (* n (factorial (- n 1)))))
```

↕

```
function factorial(n) {
  if (n == 0) {
    return 1;
  } else {
    return n * factorial(n - 1);
  }
}
```

Výhody

- Jednoduchá syntaxe s málo pravidly → dá se rychle pochopit
- Kód se dá přímo reprezentovat v jazyce jako stromová struktura → snadné **metaprogramování**
- Je jednodušší psát externí programy pro práci s kódem (interpreter, syntaktické zvýrazňování, automatické odsazování, etc.) a předělat je na práci s jiným dialektem
- Kód se dá reprezentovat stejným způsobem jako data
- Umožňuje využívat v identifikátorech znaky, které se běžně používají jen pro operátory: `int->float`

Nevýhody

- Jelikož vše vypadá stejně, je obtížnější rozeznávat různé struktury
- Snadno se jde ztratit v závorkách

M-výrazy

Původně měl LISP vypadat nějak takto:

```
defun[factorial, (n),
  if[=[n, 0],
    1,
    *[n, factorial[-[n, 1]]]
  ]
]
```

Dnes se podobná syntaxe využívá v jazyce MATHEMATICA.

Wisp

```
display "Hello World!"
define : factorial n
    if : zero? n
       . 1
       * n : factorial {n - 1}
```

↕

```
(display "Hello World!")
(define (factorial n)
    (if (zero? n)
       1
       (* n (factorial {n - 1}))))
```

Nástroje

GNU Emacs

GNU Emacs je textový editor programovatelný ve vlastním jazyce **Emacs Lisp**.

Kromě editování textu má také spoustu dalších funkcionalit: plánování projektů, čtení e-mailu a zpráv, kalendář, …

Dřiv byla „válka editorů“ mezi uživeteli GNU Emacs a Vim.

Čeho je to zkratka?

- **E**MACS **M**eans **A** **C**rappy **S**creen
- **E**MACS **M**akes **A**ny **C**omputer **S**low
- **E**MACS **M**anuals **A**re **C**ryptic & **S**urreal
- **E**ven a **M**aster of **A**rts **C**omes **S**impler
- **E**MACS **M**anuals **A**lways **C**ause **S**enility
- **E**ventually **M**unches **A**ll **C**omputer **S**torage
- **E**ight **M**egabytes **A**nd **C**onstantly **S**wapping
- **E**lsewhere **M**aybe **A**ll **C**ommands are **S**imple
- **G**enerally **N**ot **U**sed, **E**xcept by **M**iddle-**A**ged **C**omputer **S**cientists
- **E**MACS **M**akers **A**re **C**razy **S**ickos
- **E**very **M**ale **A**dolescent **C**raves **S**ex

Ve skutečnosti: **E**DITING **MAC**RO**S**

Paredit

Paredit je rozšíření pro GNU Emacs, které přidává příkazy pro snadnou manipulaci s S-výrazy.

Umožňuje například jednou klávesovou zkratkou rozdělit výraz do více řádek, zredukovat výraz na jeho první argument, etc.

Parinfer

Paredit je rozšíření pro GNU Emacs, které automaticky přidává a odebírá závorky u S-výrazů na základě změny odsazení.

Jedná se tedy o jakýsi kompromis mezi S-výrazy a jazyky s odsazovací syntaxí jako Python nebo Haskell.

Dialekty

Dialekty

Původní Lisp se dnes již nepoužívá, jelikož je pevně svázán s architekturou IBM 704.

Místo něj v prbbéhu historia vznikla spousta více či méně podobných jazyků.

Je jich opravdu hodně, ukážu jen ty nejvýznamnější, které se dnes stále používají.

Common Lisp

- Začal jako sjednocení několika různých dialektů, které se v historii vytvořily
- Má přes deset různých implementací, žádná není oficiální
- Podporuje procedurální, funkcionální a objektové programování a metaprogramování
- Standardizován institucí ANSI

Scheme

- Vytvořen v roce 1970 na MIT
- První dialekt Lispu s lexikálním rozsahem proměnných a eliminací koncového volání
- Minimalistický → snadný na implementaci
- Na rozdíl od Common Lisp zachází s funkcemi jako s proměnnými
- Standardizován institucí IEEE, standard R^{7}RS (**R**evised^7 **R**eport on **S**cheme)

Clojure

- Vytvořen pro **Java Virtual Machine**, nyní běží i na jiných platformách
- Zdůrazňuje používání neměnných objektů
- Umožňuje snadnou spolupráci s Javou
- Podmnožina Clojure se dá používat jako datový formát (Extensible Data Notation)

PicoLisp

- Vytvořen v roce 1988
- Velmi jednoduchý datový model (všechno je buňka)
- Nerozlišuje mezi makry a funkcemi
- Umí jednoduše spolupracovat s C
- Nepodporuje čísla s plovoucí desetinnou čárkou nebo pole

Janet

- Jako primární datovou strukturu používá **pole**, nikoliv spojové seznamy
- Dá se snadno vnořit do programu v C (stejně jako některé implementace jiných Lispů)
- Optimalizace koncového volání
- Zabudovaná podpora pro Parsing Expression Grammars

Hy

- **Python** ve formě S-výrazů s podporou maker
- Inspirovaný Clojure
- Nesnaží se být minimalistický
- Existují také Lisp verze dalších jazyků: Lua → Fennel, JavaScript → Sibilant, PHP → Phel

Racket

- „Jazykově orientované programování“
- Zabudované vývojové prostření: DrRacket

Jazyky inspirované Lispem

- Python
- JavaScript
- Lua
- Ruby
- Mathematica
- Haskell
- Perl
- R
- Scala
- Nim

Temné stránky

car a cdr

Funkce `car` vrací začátek seznamu / levou část uspořádané dvojice: `(car '(2 3 5 7)) = 2`.

Funkce `cdr` vrací zbytek seznamu / pravou část uspořádané dvojice: `(cdr '(2 3 5 7)) = '(3 5 7)`.

Názvy pochází z kódů pro registry na IBM 704.

Existují různé šílené kombinace:

```
(cddadr x) = (cdr (cdr (car (cdr x))))
```

Není rovnost jako rovnost

Programátoři: *jsou zmatení, že JavaScript má dva různé druhy rovnosti (`==`, `===`)*
Lisp: Hold my parentheses.

`=`, `eq`, `eql`, `equal`, `equalp`, `string-equal`, `char-equal`, `tree-equal`, …

Liší se v tom, jestli testují stejné místo v paměti, rovnost typů, číselnou rovnost, strukturální rovnost, …

Dynamické/interpretované jazyky obecně

Běh programu je obecně pomalejší než u kompilovaných jazyků a není možné předem kontrolovat typové chyby.

Některé dialekty Lispu (např. Common Lisp) umožňují přidávat typové anotace a kompilovat do strojového kódu, což tyto problémy do určité míry zlepšuje.

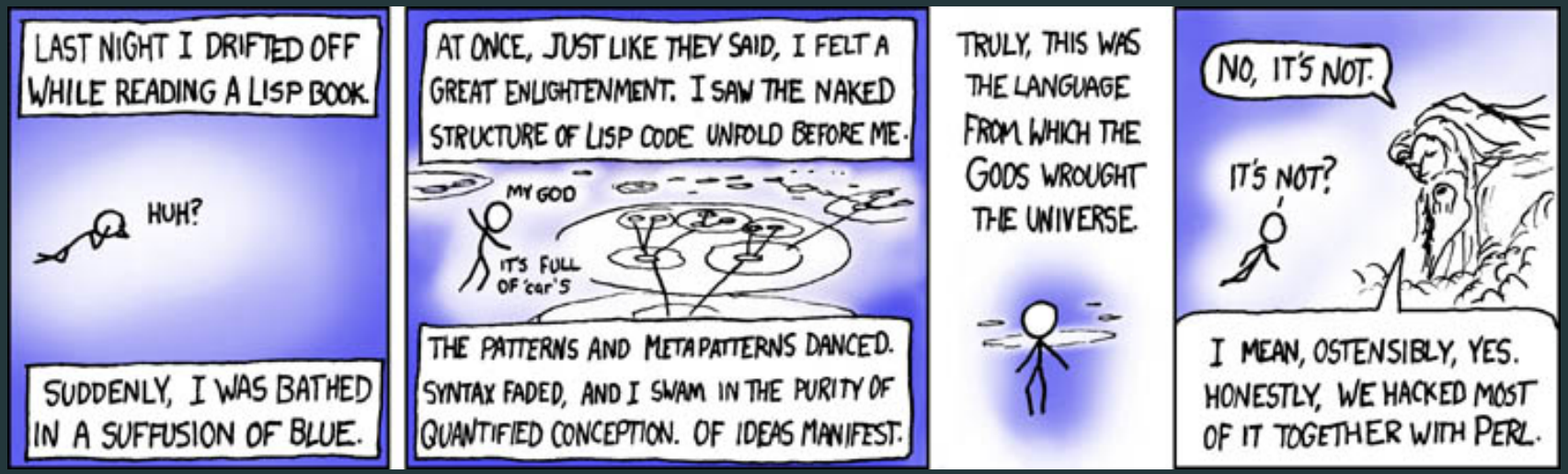
Bonus

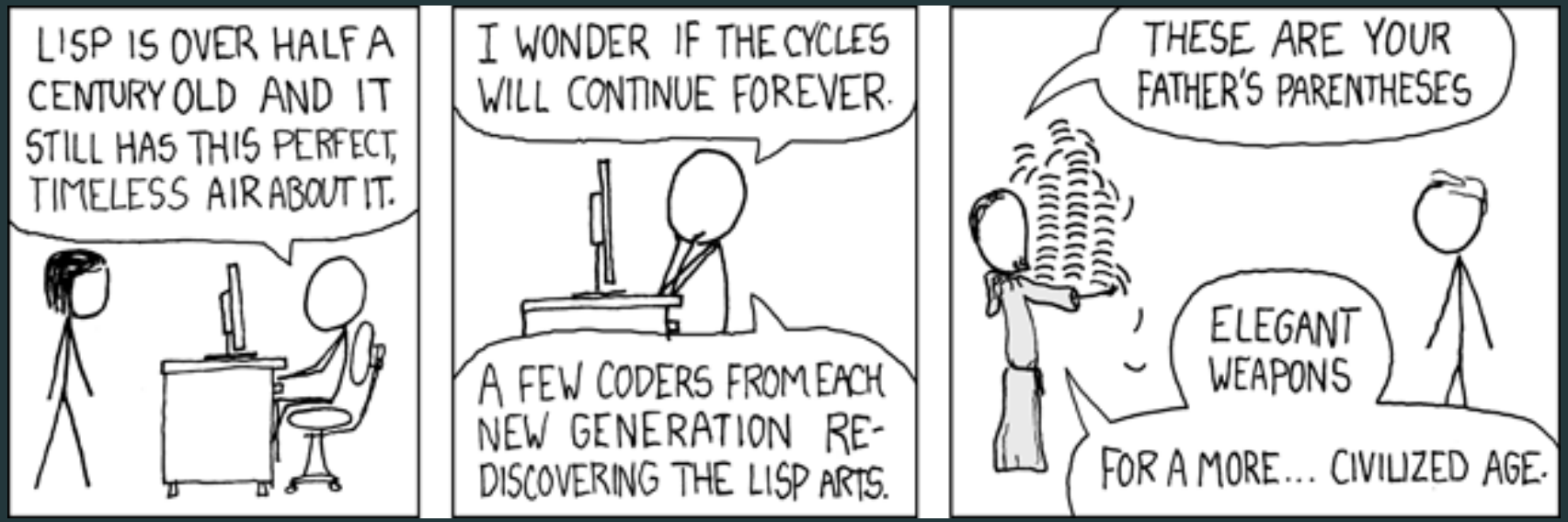
XML

```
<svg xmlns="http://www.w3.org/2000/svg"
  viewBox="0 0 5 5" width="480" height="480">
  <rect width="5" height="5" fill="#000"/>
  <path d="M 1 1 L 1 4 L 4 4 L 4 1"
    fill="#ffd43b"/>
</svg>
```

↕

```
(svg :xmlns "http://www.w3.org/2000/svg"
  :view-box '(0 0 5 5) :width 480 :height 480
  (rect :width 5 :height 5 :fill "#000")
  (path :d '(:M 1 1 :L 1 4 :L 4 4 :L 4 1)
    :fill "#ffd43b"))
```


LAST NIGHT I DRIFTED OFF WHILE READING A LISP BOOK.
HUH?
SUDDENLY, I WAS BATHED IN A SUFFUSION OF BLUE.
AT ONCE, JUST LIKE THEY SAID, I FELT A GREAT ENLIGHTENMENT. I SAW THE NAKED STRUCTURE OF LISP CODE UNFOLD BEFORE ME.
MY GOD
IT'S FULL OF 'car'S
THE PATTERNS AND METAPATTERNS DANCED. SYNTAX FADED, AND I SWAM IN THE PURITY OF QUANTIFIED CONCEPTION. OF IDEAS MANIFEST.
TRULY, THIS WAS THE LANGUAGE FROM WHICH THE GODS WROUGHT THE UNIVERSE.
NO, IT'S NOT.
IT'S NOT?
I MEAN, OSTENSIBLY, YES. HONESTLY, WE HACKED MOST OF IT TOGETHER WITH PERL.


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.

Děkuji za pozornost