Neprogramátor

... že programování je čarování, že programátoři zvládnou udělat věci, které normální lidé nezvládnou...

(Aaron, The Internet's Own Boy)

Český jazyk

Normálním lidem jde komunikovat. Mají na to jazyk.

- podstatná jména: "kniha"
- přídavná jména: "kouzelná"
- číslovky: 0 3.14 -9
- slovesa: sečti spoj
- ...

velké písmeno → slovosled → . ! ?

↑ syntax ↑ ↓ sémantika ↓

Význam slov a vět se učíme od dětství a chápeme intuitivně.

Sečti 0 a 3.14 a -9!

Spoj "kouzelná" a "kniha".

Sečti 0 a "kniha".

Kouzelný jazyk Racket

"kniha"
"kouzelná" } ukrývají *informaci*
0 3.14 -9

string-append } s podstatnými jmény
+ } sloveso říká *co*

(sloveso podstatná jména)

↑ syntax ↑ ↓ sémantika ↓

Význam slov a vět je přesně daný specifikací, ale odhadnutelný!

```
1 (+ 0 3.14 -9)
2 (string-append "kouzelná" "kniha")
3 (+ 0 "kniha")
```

Kouzelná hůlka

... *vykoná* kouzelnou větu tak, že z kouzelné věty udělá kouzelné slovo.

Výsledky vykonání

```
1 -5.859999999999999
2 "kouzelnákniha"

3 . . +: contract violation
    expected: number?
    given: "kniha"

13 #f

15 #t
16 #t
17 #t
18 #t

24 18

25 "1.|2."

32 "1.|2."
```

Podmínkové výrazy

```
25 (if 1.větší                32 (cond
26     (if 1.delší            33     ((and 1.větší
27         "vždy 1."          34           1.delší)
28         "1.|2.")           35      "vždy 1.")
29     (if 1.delší            36     ((or 1.větší
30         "1.|2."            37          1.delší)
31         "vždy 2."))        38      "1.|2.")
                              39     (else "vždy 2.")))
```


Pravdivostní tabulka

	#t	#f
#t	"vždy 1."	"1.|2."
#f	"1.|2."	"vždy 2."

nové sloveso	podstatná jména
definice funkce	*primitivní datové typy*

```
10 (define (1.delší? 1.slovo 2.slovo)
11   (> (string-length 1.slovo)
12      (string-length 2.slovo)))
13 (1.delší? "kniha" "kouzelná")
```

seznam	slovesa			
podstatná jména spojená do jednoho	+ - * / sqrt expt	*čísla*	*text*	*logika*
	modulo round abs	0 3.14 -9	"kniha" "3.14"	#t #f
	sin max log atan	predikáty	substring	and
	= < > <= >= odd?	výsledek vykonání je #t nebo #f	string->number	or
	string=? equal?		number->string	not

```
14 (define čísla (list 0 3.14 -9))
15 (equal? čísla '(0 3.14 -9))
16 (equal? 0 (car čísla))
17 (equal? '(3.14 -9) (cdr čísla))
18 (equal? čísla ; (cons 0 '(3.14 -9))
19          (cons (car čísla) (cdr čísla)))
```

rekurze	pojmenování informací
použití funkce při její definici	definice proměnných

```
20 (define (sečti od až) ; včetně
21   (if (= od až)
22       až ; základní krok / indukční krok
23       (+ od (sečti (+ 1 od) až))))
24 (sečti 3 6)
```

```
4 (define 1.číslo 3.14)
5 (define 2.číslo -9)
6 (define 1.větší (> 1.číslo 2.číslo))
7 (define 1.delší
8   (> (string-length "kniha")
9      (string-length "kouzelná")))
```

Kouzelné čtení

Jazyk je jeden ze způsobů, jak se dají sdílet myšlenky. Použije se při mluvení a poslouchání a psaní a čtení. To, že kouzelná hůlka dokáže vykonat věty zapsané v některém z jazyků, neovlivňuje možnost sdílet v jazyce myšlenky. Vykonání kouzelné věty kouzelnou hůlkou ale souvisí s tím, jak se dá s myšlenkami vyjádřenými v kouzelném jazyce pracovat, protože dost práce udělá právě kouzelná hůlka.

Krátká matematická myšlenka

```
(define (Pythagorova-věta a b c)
  ; a, b jsou odvěsny, c je přepona
  (= (* c c) (+ (* a a) (* b b))))
(define (vypočítej-přeponu-pro-odvěsny a b)
  (sqrt (+ (* a a) (* b b))))
(define (vypočítej-odvěsnu-pro-přeponu-a-odvěsnu c a)
  (sqrt (- (* c c) (* a a))))
```

Krátká chemická myšlenka

```
(define (oxidační-číslo->koncovka oxidační-číslo)
  (cond ((= 1 oxidační-číslo) "-ný")
        ((= 2 oxidační-číslo) "-natý")
        ((= 3 oxidační-číslo) "-itý")
        ((= 4 oxidační-číslo) "-ičitý")
        ((= 5 oxidační-číslo) "-ičný, -ečný")
        ((= 6 oxidační-číslo) "-ový")
        ((= 7 oxidační-číslo) "-istý")
        ((= 8 oxidační-číslo) "-ičelý")
        ((= 9 oxidační-číslo) "-utý")))
```

Několik myšlenek o logice

```
; základní definice a opak (negace) jedné věty
(define pravda #t)
(define (je-pravda? věta)
  (if (equal? pravda věta)
      pravda
      lež))
(define lež #f)
; Z hlediska logiky je nekorektní
; používat lež jako opak pravdy.
(define (je-lež? věta)
  (if (equal? lež věta)
      pravda
      lež))
(define (neplatí-že věta)
  (cond ((je-pravda? věta) lež)
        ((je-lež? věta) pravda)))

; možné kombinace dvou vět
(define (platí-obě? první-věta druhá-věta)
  (if (je-pravda? první-věta)
      (if (je-pravda? druhá-věta)
          pravda
          lež)
      lež))
(define (platí-první-a-druhá-ne? první-věta druhá-věta)
  (if (je-pravda? první-věta)
      (if (je-lež? druhá-věta)
          pravda
          lež)
      lež))
(define (platí-druhá-a-první-ne? první-věta druhá-věta)
  (if (je-pravda? druhá-věta)
      (if (je-lež? první-věta)
          pravda
          lež)
      lež))
(define (neplatí-ani-jedna? první-věta druhá-věta)
  (if (je-lež? první-věta)
      (if (je-lež? druhá-věta)
          pravda
          lež)
      lež))
(define (řekni-jestli-platí-že má-být věta)
  (cond ((platí-obě? má-být věta) "platí")
        ((neplatí-ani-jedna? má-být věta) "platí")
        (else "neplatí")))

; souvětí dvou vět
(define (souvětí-s-a první-věta druhá-věta)
  (cond
    ((platí-obě? první-věta druhá-věta) pravda)
    ((platí-první-a-druhá-ne? první-věta druhá-věta) lež)
    ((platí-druhá-a-první-ne? první-věta druhá-věta) lež)
    ((neplatí-ani-jedna? první-věta druhá-věta) lež)))
(define (souvětí-s-nebo první-věta druhá-věta)
  (cond
    ((platí-obě? první-věta druhá-věta) pravda)
    ((platí-první-a-druhá-ne? první-věta druhá-věta) pravda)
    ((platí-druhá-a-první-ne? první-věta druhá-věta) pravda)
    ((neplatí-ani-jedna? první-věta druhá-věta) lež)))
(define (souvětí-s-vylučovacím-nebo první-věta druhá-věta)
  (cond
    ((platí-obě? první-věta druhá-věta) lež)
    ((platí-první-a-druhá-ne? první-věta druhá-věta) pravda)
    ((platí-druhá-a-první-ne? první-věta druhá-věta) pravda)
    ((neplatí-ani-jedna? první-věta druhá-věta) lež)))
(define (když první-věta druhá-věta)
  (cond
    ((platí-obě? první-věta druhá-věta) pravda)
    ((platí-první-a-druhá-ne? první-věta druhá-věta) lež)
    ((platí-druhá-a-první-ne? první-věta druhá-věta) pravda)
    ((neplatí-ani-jedna? první-věta druhá-věta) pravda)))
(define (právě-tehdy-když první-věta druhá-věta)
  (cond
    ((platí-obě? první-věta druhá-věta) pravda)
    ((platí-první-a-druhá-ne? první-věta druhá-věta) lež)
    ((platí-druhá-a-první-ne? první-věta druhá-věta) lež)
    ((neplatí-ani-jedna? první-věta druhá-věta) pravda)))

; odborné názvy
(define konjunkce souvětí-s-a)
(define disjunkce souvětí-s-nebo)
(define negace neplatí-že)
(define implikace když)
(define ekvivalence právě-tehdy-když)

; Jaká je negace věty 'Když prší je mokro'?
(define prší pravda)
(define je-mokro pravda)
(řekni-jestli-platí-že
 (neplatí-že (když prší je-mokro))
 (když (neplatí-že prší) je-mokro))
 ; Když neprší je mokro.
(řekni-jestli-platí-že
 (neplatí-že (když prší je-mokro))
 (když prší (neplatí-že je-mokro)))
 ; Když prší není mokro.
(řekni-jestli-platí-že
 (neplatí-že (když prší je-mokro))
 (když (neplatí-že prší) (neplatí-že je-mokro)))
 ; Když neprší není mokro.
```