Paradigmata programování 1

Přednáška 1. Symbolické výrazy

Michal Krupka

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Outline

Úvod

- Programovací jazyk Scheme
- Vývojové prostředí DrRacket
- Principy programování

Dávat důraz na principy mimo jiné znamená

- Nepostupovat metodou pokus/omyl
- Vědět vždy přesně, co se v počítači děje...
- ...až do té míry, že budeme umět předpovídat výsledky.

...Proto budeme u zkoušky programovat na papír.

Úvod

- Programovací jazyk Scheme
- Vývojové prostředí DrRacket
- Principy programování

Dávat důraz na principy mimo jiné znamená

- Nepostupovat metodou pokus/omyl
- Vědět vždy přesně, co se v počítači děje...
- ...až do té míry, že budeme umět předpovídat výsledky.

...Proto budeme u zkoušky programovat na papír.

Úvod

- Programovací jazyk Scheme
- Vývojové prostředí DrRacket
- Principy programování

Dávat důraz na principy mimo jiné znamená

- Nepostupovat metodou pokus/omyl
- Vědět vždy přesně, co se v počítači děje...
- ...až do té míry, že budeme umět předpovídat výsledky.

...Proto budeme u zkoušky programovat na papír.

Outline

Outline

Symbolické výrazy

Symbolický výraz je

- jednoduchý výraz, neboli *atom*, nebo
- složený výraz, neboli *seznam*.

Atom je

- číslo nebo
- symbol nebo
- ...

Číslo: `10`, `-5`, `0+1i`, `2/3`, `#i0.4999999999999999`, `#i1.2246467991473532e-16`.

Symbol: `pi`, `r`, `a1`, `sqrt`, `+`, `/`, `1+1`.

Symbolické výrazy

Symbolický výraz je

- jednoduchý výraz, neboli *atom*, nebo
- složený výraz, neboli *seznam*.

Atom je

- číslo nebo
- symbol nebo
- ...

Číslo: `10`, `-5`, `0+1i`, `2/3`, `#i0.4999999999999999`, `#i1.2246467991473532e-16`.

Symbol: `pi`, `r`, `a1`, `sqrt`, `+`, `/`, `1+1`.

Symbolické výrazy

Symbolický výraz je

- jednoduchý výraz, neboli *atom*, nebo
- složený výraz, neboli *seznam*.

Atom je

- číslo nebo
- symbol nebo
- ...

Číslo: `10`, `-5`, `0+1i`, `2/3`, `#i0.4999999999999999`, `#i1.2246467991473532e-16`.

Symbol: `pi`, `r`, `a1`, `sqrt`, `+`, `/`, `1+1`.

Symbolické výrazy

Symbolický výraz je

- jednoduchý výraz, neboli *atom*, nebo
- složený výraz, neboli *seznam*.

Atom je

- číslo nebo
- symbol nebo
- ...

Číslo: `10`, `-5`, `0+1i`, `2/3`, `#i0.4999999999999999`, `#i1.2246467991473532e-16`.

Symbol: `pi`, `r`, `a1`, `sqrt`, `+`, `/`, `1+1`.

Symbolické výrazy

Seznam (složený výraz) je posloupnost jednoho nebo více výrazů. Prvky seznamu jsou odděleny mezerami nebo jinými prázdnými znaky. Seznam začíná levou a končí pravou kulatou závorkou.

Příklady seznamů:

- `(+ 1 2)`
- `(+)`
- `(1 2 3)`
- `(1 + 2 + 3)`
- ```
  (* (sin (/ pi 4))
     (sin (/ pi 4)))
  ```
  ```

Symbolické výrazy

Seznam (složený výraz) je posloupnost jednoho nebo více výrazů. Prvky seznamu jsou odděleny mezerami nebo jinými prázdnými znaky. Seznam začíná levou a končí pravou kulatou závorkou.

Příklady seznamů:

- `(+ 1 2)`
- `(+)`
- `(1 2 3)`
- `(1 + 2 + 3)`
-
```
(* (sin (/ pi 4))
   (sin (/ pi 4)))
```

Symbolické výrazy

Toto nejsou symbolické výrazy:

- `)`
- `(+ (- 10 11)`
- `ah oj`

Vytváření symbolických výrazů

První symbol v seznamu označuje název matematického výrazu (součet, rozdíl, součin, podíl). Označuje tedy poslední operaci prováděnou ve výpočtu.

Například výraz

$$\frac{5-3}{5+3}$$

```
(/ (- 5 3)
   (+ 5 3))
```

Pro přehlednost:

Vytváření symbolických výrazů

První symbol v seznamu označuje název matematického výrazu (součet, rozdíl, součin, podíl). Označuje tedy poslední operaci prováděnou ve výpočtu.

Například výraz

$$\frac{5-3}{5+3}$$

Pro přehlednost:

```
(/ (- 5 3)
   (+ 5 3))
```

Vytváření symbolických výrazů

První symbol v seznamu označuje název matematického výrazu (součet, rozdíl, součin, podíl). Označuje tedy poslední operaci prováděnou ve výpočtu.

Například výraz

$$\frac{5-3}{5+3}$$

je podíl:

```
(/                    )
```

Pro přehlednost:

```
(/ (- 5 3)
   (+ 5 3))
```

Vytváření symbolických výrazů

První symbol v seznamu označuje název matematického výrazu (součet, rozdíl, součin, podíl). Označuje tedy poslední operaci prováděnou ve výpočtu.

Například výraz

$$\frac{5-3}{5+3}$$

je podíl rozdílu:

```
(/ (- 5 3)          )
```

Pro přehlednost:

```
(/ (- 5 3)
   (+ 5 3))
```

Vytváření symbolických výrazů

První symbol v seznamu označuje název matematického výrazu (součet, rozdíl, součin, podíl). Označuje tedy poslední operaci prováděnou ve výpočtu.

Například výraz

$$\frac{5-3}{5+3}$$

je podíl rozdílu a součtu:

```
(/ (- 5 3) (+ 5 3))
```

Pro přehlednost:

```
(/ (- 5 3)
   (+ 5 3))
```

Vytváření symbolických výrazů

První symbol v seznamu označuje název matematického výrazu (součet, rozdíl, součin, podíl). Označuje tedy poslední operaci prováděnou ve výpočtu.

Například výraz

$$\frac{5-3}{5+3}$$

je podíl rozdílu a součtu:

```
(/ (- 5 3) (+ 5 3))
```

Pro přehlednost:

```
(/ (- 5 3)
   (+ 5 3))
```

Vytváření symbolických výrazů

Výraz

$$\sin\left(\frac{5-3}{5+3}\cdot\pi\right)$$

je sinus:

```
(sin
          )
```

Vytváření symbolických výrazů

Výraz

$$\sin\left(\frac{5-3}{5+3}\cdot\pi\right)$$

je sinus součinu:

```
(sin (*
          ))
```

Vytváření symbolických výrazů

Výraz

$$\sin\left(\frac{5-3}{5+3}\cdot\pi\right)$$

je sinus součinu dříve uvedeného podílu:

```
(sin (* (/ (- 5 3) (+ 5 3))
        ))
```

Vytváření symbolických výrazů

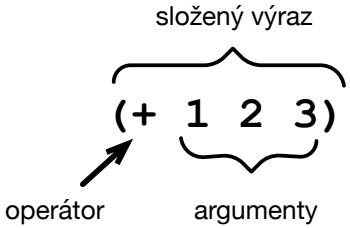
Výraz

$$\sin\left(\frac{5-3}{5+3}\cdot\pi\right)$$

je sinus součinu dříve uvedeného podílu a čísla π:

```
(sin (* (/ (- 5 3) (+ 5 3))
        pi))
```

Terminologie pro složené výrazy (seznamy)



Operátor: *co* se má udělat
Argumenty: *s čím* se to má udělat

(Tedy: „sečti čísla 1, 2 a 3“)

Prefixová notace

Scheme důsledně používá tzv. prefixovou notaci: operátor je vždy uveden *před* argumenty. To je velké zjednodušení proti ostatním programovacím jazykům i matematice:

prefix: $\sin x$, -5

infix: $x + y \cdot z - 3$ (komplikace: přednost operací)

postfix: $10!$

a další: x^y, $\sqrt[3]{10}$, $\frac{5}{6}$, $\bar{z}$, $|z|$, $\int_a^b 2x \, \mathrm{d}x$

Na důsledně prefixovou notaci je třeba si ale zvyknout.

Prefixová notace

Scheme důsledně používá tzv. prefixovou notaci: operátor je vždy uveden *před* argumenty. To je velké zjednodušení proti ostatním programovacím jazykům i matematice:

prefix: $\sin x,\ -5$

infix: $x + y \cdot z - 3$ (komplikace: přednost operací)

postfix: $10!$

a další: $x^y,\ \sqrt[3]{10},\ \frac{5}{6},\ \bar{z},\ |z|,\ \int_a^b 2x\,\mathrm{d}x$

Na důsledně prefixovou notaci je třeba si ale zvyknout.

Outline

Vyhodnocovací proces

Číslo $\frac{1}{4}$ je hodnotou symbolického výrazu `(/ (- 5 3) (+ 5 3))`.

Zjednodušený vyhodnocovací proces:

Vyhodnocení výrazu E

Je-li E symbol, výsledkem je hodnota symbolu E.

Je-li E jiný atom než symbol, výsledkem je E.

Je-li E seznam s operátorem o a argumenty $a_1 \dots a_n$, pak

1. se zjistí hodnota p operátoru o (opět vyhodnocovacím procesem), p musí být procedura,
2. zjistí se hodnoty $v_1 \dots v_n$ argumentů $a_1 \dots a_n$ (opět vyhodnocovacím procesem).
3. Výsledkem je výsledek aplikace procedury p na hodnoty $v_1 \dots v_n$.

Vyhodnocovací proces

Číslo $\frac{1}{4}$ je hodnotou symbolického výrazu `(/ (- 5 3) (+ 5 3))`.

Zjednodušený vyhodnocovací proces:

Vyhodnocení výrazu E

Je-li E symbol, výsledkem je hodnota symbolu E.

Je-li E jiný atom než symbol, výsledkem je E.

Je-li E seznam s operátorem o a argumenty $a_1 \dots a_n$, pak

1. se zjistí hodnota p operátoru o (opět vyhodnocovacím procesem), p musí být procedura,
2. zjistí se hodnoty $v_1 \dots v_n$ argumentů $a_1 \dots a_n$ (opět vyhodnocovacím procesem).
3. Výsledkem je výsledek aplikace procedury p na hodnoty $v_1 \dots v_n$.

Hodnota symbolu

Symbol: jméno hodnoty. Například:

```
> pi
#i3.141592653589793
```

Hodnotou symbolu `pi` je číslo `#i3.141592653589793`. Proto:

```
> (* 2 pi)
#i6.283185307179586
```

Symbolu, který má hodnotu, také říkáme proměnná.

```
> ahoj
ahoj: this variable is not defined
```

Symbol `ahoj` nemá hodnotu. (Není to proměnná.)

Hodnota symbolu

Symbol: jméno hodnoty. Například:

```
> pi
#i3.141592653589793
```

Hodnotou symbolu `pi` je číslo `#i3.141592653589793`. Proto:

```
> (* 2 pi)
#i6.283185307179586
```

Symbolu, který má hodnotu, také říkáme proměnná.

```
> ahoj
ahoj: this variable is not defined
```

Symbol `ahoj` nemá hodnotu. (Není to proměnná.)

Procedury

Už jsme se setkali i s jinými symboly, které mají hodnotu:

```
> +
#<procedure:+>
> /
#<procedure:/>
```

Jejich hodnotou jsou procedury.

Procedura je prvek programu, který umožňuje provádět nějaké (například matematické) výpočty se *vstupními hodnotami*. Výpočet se spustí tzv. *aplikací procedury na vstupní hodnoty*. O výsledku výpočtu někdy říkáme, že ho procedura *vrací*.

Procedury

Už jsme se setkali i s jinými symboly, které mají hodnotu:

```
> +
#<procedure:+>
> /
#<procedure:/>
```

Jejich hodnotou jsou procedury.

Procedura je prvek programu, který umožňuje provádět nějaké (například matematické) výpočty se *vstupními hodnotami*. Výpočet se spustí tzv. *aplikací procedury na vstupní hodnoty*. O výsledku výpočtu někdy říkáme, že ho procedura *vrací*.

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

- Vyhodnocení výrazu `/`
  Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

- Vyhodnocení výrazu `(- 5 3)`
  Je to seznam, vyhodnotí se nejprve operátor:
  - Vyhodnocení výrazu `-`
    Je to symbol, výsledkem je jeho hodnota `#<procedure:->`
  - pak se vyhodnotí argumenty:
    - Vyhodnocení výrazu `5`
      Je to číslo, výsledkem je číslo 5
    - Vyhodnocení výrazu `3`
      Je to číslo, výsledkem je číslo 3
  - Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

- Vyhodnocení výrazu `/`
  Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

- Vyhodnocení výrazu `(- 5 3)`
  Je to seznam, vyhodnotí se nejprve operátor:
  - Vyhodnocení výrazu `-`
    Je to symbol, výsledkem je jeho hodnota `#<procedure:->`
  - pak se vyhodnotí argumenty:
    - Vyhodnocení výrazu `5`
      Je to číslo, výsledkem je číslo 5
    - Vyhodnocení výrazu `3`
      Je to číslo, výsledkem je číslo 3
  - Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

- Vyhodnocení výrazu `/`
  - Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

- Vyhodnocení výrazu `(- 5 3)`
  - Je to seznam, vyhodnotí se nejprve operátor:
    - Vyhodnocení výrazu `-`
      - Je to symbol, výsledkem je jeho hodnota `#<procedure:->`
  - pak se vyhodnotí argumenty:
    - Vyhodnocení výrazu `5`
      - Je to číslo, výsledkem je číslo 5
    - Vyhodnocení výrazu `3`
      - Je to číslo, výsledkem je číslo 3
  - Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu /

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu -

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu /

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu -

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu /

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu –

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

> Vyhodnocení výrazu `/`
>
> Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

> Vyhodnocení výrazu `(- 5 3)`
>
> Je to seznam, vyhodnotí se nejprve operátor:
>
> > Vyhodnocení výrazu `-`
> >
> > Je to symbol, výsledkem je jeho hodnota `#<procedure:->`
>
> pak se vyhodnotí argumenty:
>
> > Vyhodnocení výrazu `5`
> >
> > Je to číslo, výsledkem je číslo 5
> >
> > Vyhodnocení výrazu `3`
> >
> > Je to číslo, výsledkem je číslo 3
>
> Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `/`

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `-`

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `5`

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu `3`

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `/`

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `-`

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `5`

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu `3`

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu /

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu –

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `/`

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `-`

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `5`

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu `3`

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu /

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu –

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `/`

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `-`

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `5`

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu `3`

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `/`

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `-`

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad

Vyhodnocení výrazu `(/ (- 5 3) (+ 5 3))`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `/`

Je to symbol, výsledkem je jeho hodnota `#<procedure:/>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu `(- 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu `-`

Je to symbol, výsledkem je jeho hodnota `#<procedure:->`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:->` na 5 a 3, tedy číslo 2

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy 1/4

Vyhodnocovací proces: příklad (pokračování)

(vyhodnocujeme výraz `(/ (- 5 3) (+ 5 3))`)

Vyhodnocení výrazu `(+ 5 3)`

Je to seznam, vyhodnotí se nejprve operátor:

Vyhodnocení výrazu +

Je to symbol, výsledkem je jeho hodnota `#<procedure:+>`

pak se vyhodnotí argumenty:

Vyhodnocení výrazu 5

Je to číslo, výsledkem je číslo 5

Vyhodnocení výrazu 3

Je to číslo, výsledkem je číslo 3

Výsledkem je aplikace procedury `#<procedure:+>` na hodnoty 5 a 3, tedy 8

Výsledkem je aplikace procedury `#<procedure:/>` na hodnoty 2 a 8, tedy `1/4`

Outline

Outline

Speciální operátory `if` a `define`

Podmíněný výraz je složený výraz s operátorem `if`. Takový výraz musí mít přesně tři argumenty (jinak dojde k chybě). Vyhodnocuje se takto:

Vyhodnocení výrazu (`if` a b c)

1. Vyhodnotí se a na hodnotu u.
2. Pokud je u rovno `#false`, vyhodnotí se c a vrátí jeho hodnota.
3. Pokud není, vyhodnotí se b a vrátí jeho hodnota.

Výraz se speciálním operátorem `define` musí mít dva argumenty, z nichž první musí být symbol, který zatím není definován jako jméno hodnoty (proměnná). Operátor tuto definici provede a hodnotu nastaví.

Vyhodnocení výrazu (`define` a b)

1. Vyhodnotí se b.
2. symbol a se učiní jménem této hodnoty.

Vyhodnocovací proces (znovu)

Vyhodnocení výrazu E

Je-li E symbol, výsledkem je hodnota symbolu E.

Je-li E jiný atom než symbol, výsledkem je E.

Je-li E seznam s operátorem o a argumenty $a_1 \dots a_n$, pak

Jestliže o je speciální operátor, seznam se vyhodnotí podle pravidel tohoto speciálního operátoru.

Jinak
1. se zjistí hodnota p operátoru o (opět vyhodnocovacím procesem), p musí být procedura,
2. zjistí se hodnoty $v_1 \dots v_n$ argumentů $a_1 \dots a_n$ (opět vyhodnocovacím procesem).
3. Výsledkem je výsledek aplikace procedury p na hodnoty $v_1 \dots v_n$.

Outline

Pojmy k zapamatování

Symbolický výraz, atom, seznam, číslo, symbol, logická (pravdivostní) hodnota. Symbol jako jméno hodnoty, procedura. Operátor, argumenty. Vyhodnocovací proces. Speciální operátor, speciální operátory `if` a `define`.