Paradigmata programování 2

Přednáška 10. Zásobníkové výpočty

verze z 20. dubna 2024

Michal Krupka

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Obsah

Viditelnost a životnost vazeb

(*scope* a *extent*)

Viditelnost a životnost vazeb

(*scope* a *extent*)

- Používají se k popisu vazeb symbolů (a dalších věcí).
- Určují, za jakých okolností (např. z jakých částí programu) jsou vidět a po jaký čas existují.

Viditelnost a životnost vazeb

(*scope* a *extent*)

- Používají se k popisu vazeb symbolů (a dalších věcí).
- Určují, za jakých okolností (např. z jakých částí programu) jsou vidět a po jaký čas existují.

Lexikální viditelnost: vazba symbolu je vidět pouze v určité oblasti zdrojového kódu.

Neomezenou viditelnost mají vazby, které jsou použitelné kdekoli (na libovolném místě) zdrojového kódu (opět pokud nejsou překryty jinou vazbou).

Viditelnost a životnost vazeb

(*scope* a *extent*)

- Používají se k popisu vazeb symbolů (a dalších věcí).
- Určují, za jakých okolností (např. z jakých částí programu) jsou vidět a po jaký čas existují.

Lexikální viditelnost: vazba symbolu je vidět pouze v určité oblasti zdrojového kódu.

Neomezenou viditelnost mají vazby, které jsou použitelné kdekoli (na libovolném místě) zdrojového kódu (opět pokud nejsou překryty jinou vazbou).

Viditelnost a životnost vazeb

(*scope* a *extent*)

- Používají se k popisu vazeb symbolů (a dalších věcí).
- Určují, za jakých okolností (např. z jakých částí programu) jsou vidět a po jaký čas existují.

Lexikální viditelnost: vazba symbolu je vidět pouze v určité oblasti zdrojového kódu.

Neomezenou viditelnost mají vazby, které jsou použitelné kdekoli (na libovolném místě) zdrojového kódu (opět pokud nejsou překryty jinou vazbou).

Omezenou životnost má vazba, jestliže existuje moment v čase, kdy vazba zanikne.

Neomezená životnost vazby způsobuje, že vazba po svém vzniku (z pohledu programátora) nikdy nezanikne.

Lexikální a dynamické vazby

Lexikální a dynamické vazby

Lexikální vazby. Mají lexikální viditelnost a neomezenou životnost. Jsou to vazby, které známe z Lispu.

Dynamické vazby. Mají neomezenou viditelnost a omezenou životnost. Je to starší typ, ukážeme je za chvíli.

Lexikální a dynamické vazby

Lexikální vazby. Mají lexikální viditelnost a neomezenou životnost. Jsou to vazby, které známe z Lispu.

Dynamické vazby. Mají neomezenou viditelnost a omezenou životnost. Je to starší typ, ukážeme je za chvíli.

Dále například lokální proměnné v C mají lexikální viditelnost a omezenou životnost: lokální proměnnou funkce nelze použít mimo tělo funkce a proměnná po návratu z funkce přestává existovat, což lze ověřit tím, že si zapamatujeme adresu proměnné a později se podíváme na hodnotu na adrese uloženou.
Globální proměnné mívají neomezenou viditelnost a neomezenou životnost, případně lexikální viditelnost v rámci zdrojového souboru a neomezenou životnost.

Dynamické vazby v Lispu

1. Pomocí dynamických proměnných definovaných makrem `defvar`. Všechny vazby dynamické proměnné jsou dynamické. Příklad: Proměnné `*print-length*`, `*print-base*`.

Název dynamické proměnné.

Všechny proměnné definované makrem `defvar` mají název ohraničený hvězdičkami.

2. Lokálně pomocí deklarace (nebudeme používat).

Obsah

Zásobník

Zásobník

- základní datová struktura

Zásobník

- základní datová struktura
- LIFO: operace `push` a `pop`

Zásobník

- základní datová struktura
- LIFO: operace `push` a `pop`
- používají běžící programy (procesy) v počítačích

Zásobník

- základní datová struktura
- LIFO: operace `push` a `pop`
- používají běžící programy (procesy) v počítačích
- v Lispu nejsnadněji reprezentovaný seznamem

Zásobníkový stroj

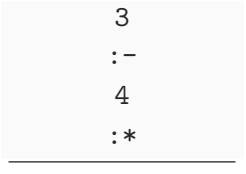
Zásobníkový stroj

rslt

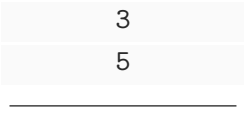
exec

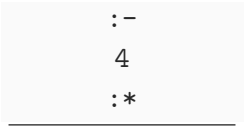
Zásobníkový stroj

Zásobníkový stroj

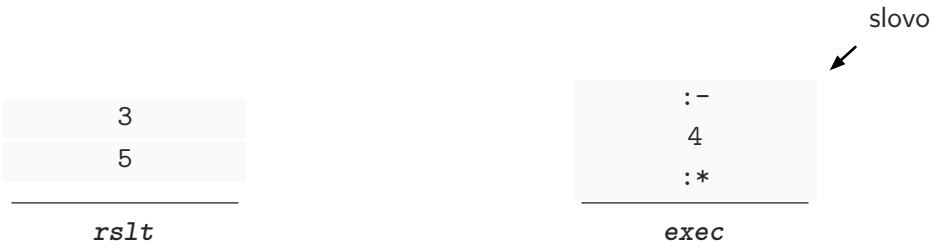


Zásobníkový stroj

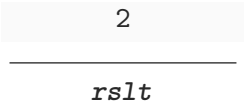


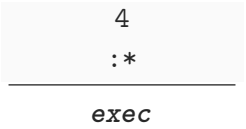


Zásobníkový stroj

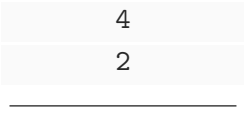


Zásobníkový stroj





Zásobníkový stroj

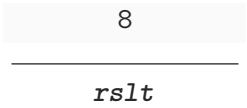


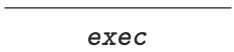
Zásobníkový stroj

Zásobníkový stroj

- Program: `5 3 :- 4 :*`

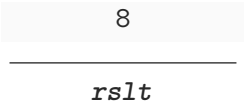
Zásobníkový stroj





- Program: `5 3 :- 4 :*`
- Vypočítal $4(5-3)$

Zásobníkový stroj



- Program: `5 3 :- 4 :*`
- Vypočítal $4(5-3)$
- *Postfixová notace*

Zásobníkový stroj

- Program: `5 3 :- 4 :*`
- Vypočítal $4(5-3)$
- *Postfixová notace*
- *Obrácená polská logika*

Základní slova

Základní slova

Manipulace se zásobníkem

`:swap`	$(a\ b - b\ a)$
`:rot`	$(a\ b\ c - b\ c\ a)$
`:drop`	$(a -)$
`:dup`	$(a - a\ a)$
`:over`	$(a\ b - a\ b\ a)$

Základní slova

Manipulace se zásobníkem

`:swap`	$(a\ b - b\ a)$
`:rot`	$(a\ b\ c - b\ c\ a)$
`:drop`	$(a\ -\ )$
`:dup`	$(a - a\ a)$
`:over`	$(a\ b - a\ b\ a)$

Aritmetické operace

`:+`	$(n_1\ n_2$ – součet $n_1 + n_2)$
`:-`	$(n_1\ n_2$ – rozdíl $n_1 - n_2)$
`:*`	$(n_1\ n_2$ – součin $n_1 \cdot n_2)$
`:/`	$(n_1\ n_2$ – podíl $n_1/n_2)$

Základní slova

Manipulace se zásobníkem

`:swap`	$(a\ b - b\ a)$
`:rot`	$(a\ b\ c - b\ c\ a)$
`:drop`	$(a\ -\ )$
`:dup`	$(a - a\ a)$
`:over`	$(a\ b - a\ b\ a)$

Aritmetické operace

`:+`	$(n_1\ n_2$ – součet $n_1 + n_2)$
`:-`	$(n_1\ n_2$ – rozdíl $n_1 - n_2)$
`:*`	$(n_1\ n_2$ – součin $n_1 \cdot n_2)$
`:/`	$(n_1\ n_2$ – podíl $n_1/n_2)$

Slova a jejich význam ukládáme na zásobník *word*. Prvky: (*slovo . význam*)

Základní slova

Manipulace se zásobníkem

`:swap`	$(a\ b - b\ a)$
`:rot`	$(a\ b\ c - b\ c\ a)$
`:drop`	$(a\ -\ )$
`:dup`	$(a - a\ a)$
`:over`	$(a\ b - a\ b\ a)$

Aritmetické operace

`:+`	$(n_1\ n_2$ – součet $n_1 + n_2)$
`:-`	$(n_1\ n_2$ – rozdíl $n_1 - n_2)$
`:*`	$(n_1\ n_2$ – součin $n_1 \cdot n_2)$
`:/`	$(n_1\ n_2$ – podíl $n_1/n_2)$

Slova a jejich význam ukládáme na zásobník *word*. Prvky: (*slovo . význam*)
Slova: vestavěná nebo uživatelská (*slovník*)

Uživatelská slova

Uživatelská slova

Slovo `:rot-` definujeme s kódem `:rot :rot`:

Uživatelská slova

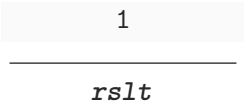
Slovo `:rot-` definujeme s kódem `:rot :rot`:

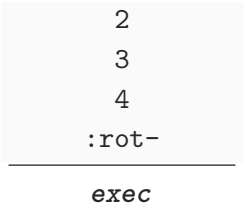

```
1
2
3
4
:rot-
```

exec

Uživatelská slova

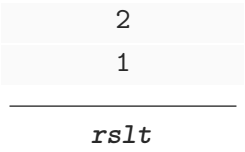
Slovo `:rot-` definujeme s kódem `:rot :rot`:

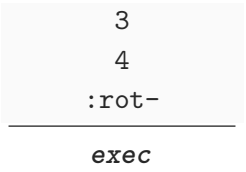




Uživatelská slova

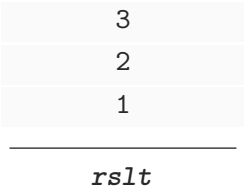
Slovo `:rot-` definujeme s kódem `:rot :rot`:

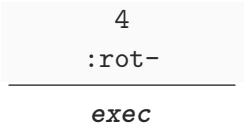




Uživatelská slova

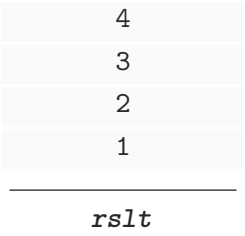
Slovo `:rot-` definujeme s kódem `:rot :rot`:

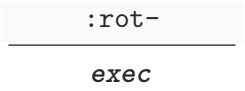




Uživatelská slova

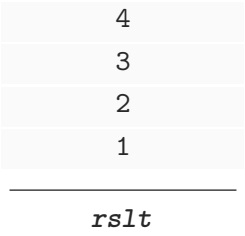
Slovo `:rot-` definujeme s kódem `:rot :rot`:

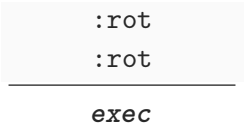




Uživatelská slova

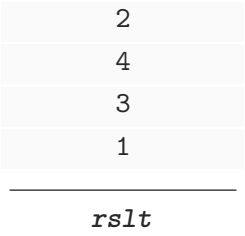
Slovo `:rot-` definujeme s kódem `:rot :rot`:

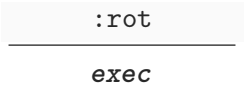




Uživatelská slova

Slovo `:rot-` definujeme s kódem `:rot :rot`:





Uživatelská slova

Slovo `:rot-` definujeme s kódem `:rot :rot`:

Další slova: větvení

Další slova: větvení

Větvení

Další slova: větvení

Větvení

```
:if ... :else ... :then
```

Další slova: větvení

Větvení

`:if ... :else ... :then`

1. Odstraní vrchol zásobníku *rslt*.

Další slova: větvení

Větvení

`:if ... :else ... :then`

1. Odstraní vrchol zásobníku *rslt*.
2. Je-li odstraněná hodnota *Pravda*, vypustí z programového zásobníku vše po následující `:else` s tím, že případné vnořené bloky `:if ... :else ... :then` správně vypouští celé.

Další slova: větvení

Větvení

`:if ... :else ... :then`

1. Odstraní vrchol zásobníku *rslt*.
2. Je-li odstraněná hodnota *Pravda*, vypustí z programového zásobníku vše po následující `:else` s tím, že případné vnořené bloky `:if ... :else ... :then` správně vypouští celé.
3. Je-li *Nepravda*, pokračuje ve vykonávání programu.

Další slova: větvení

Větvení

`:if ... :else ... :then`

1. Odstraní vrchol zásobníku *rslt*.
2. Je-li odstraněná hodnota *Pravda*, vypustí z programového zásobníku vše po následující `:else` s tím, že případné vnořené bloky `:if ... :else ... :then` správně vypouští celé.
3. Je-li *Nepravda*, pokračuje ve vykonávání programu.
4. Samostatné slovo `:else` vypustí z programového zásobníku vše po následující `:then` s tím, že případné vnořené bloky `:if ... :else ... :then` správně vypouští celé.

Další slova: větvení

Větvení

`:if ... :else ... :then`

1. Odstraní vrchol zásobníku *rslt*.
2. Je-li odstraněná hodnota *Pravda*, vypustí z programového zásobníku vše po následující `:else` s tím, že případné vnořené bloky `:if ... :else ... :then` správně vypouští celé.
3. Je-li *Nepravda*, pokračuje ve vykonávání programu.
4. Samostatné slovo `:else` vypustí z programového zásobníku vše po následující `:then` s tím, že případné vnořené bloky `:if ... :else ... :then` správně vypouští celé.
5. Samostatné slovo `:then` se ignoruje.

Zásobníkové programovací jazyky

Zásobníkové programovací jazyky

- základní: jazyk FORTH (počátky: 1968, volná inspirace pro náš jazyk)

Zásobníkové programovací jazyky

- základní: jazyk FORTH (počátky: 1968, volná inspirace pro náš jazyk)
- další: např. PostScript (laserové tiskárny)

Zásobníkové programovací jazyky

- základní: jazyk FORTH (počátky: 1968, volná inspirace pro náš jazyk)
- další: např. PostScript (laserové tiskárny)
- použití: vestavěné systémy, Open Firmware, řízení družic ...

Zásobníkové programovací jazyky

- základní: jazyk FORTH (počátky: 1968, volná inspirace pro náš jazyk)
- další: např. PostScript (laserové tiskárny)
- použití: vestavěné systémy, Open Firmware, řízení družic ...
- také runtime Javy, Pythonu, v platformě .NET

Zásobníkové programovací jazyky

- základní: jazyk FORTH (počátky: 1968, volná inspirace pro náš jazyk)
- další: např. PostScript (laserové tiskárny)
- použití: vestavěné systémy, Open Firmware, řízení družic ...
- také runtime Javy, Pythonu, v platformě .NET
- nízkoúrovňové, efektivní

Zásobníkové programovací jazyky

- základní: jazyk FORTH (počátky: 1968, volná inspirace pro náš jazyk)
- další: např. PostScript (laserové tiskárny)
- použití: vestavěné systémy, Open Firmware, řízení družic ...
- také runtime Javy, Pythonu, v platformě .NET
- nízkoúrovňové, efektivní

- pro nás dobrý příklad, protože explicitně manipulují se zásobníky

Obsah

Obsah