```
(define (sečti-rozsah od do)
  (if (= od do)
      do
      (+ od (sečti-rozsah (+ 1 od) do))))

(sečti-rozsah 3 6 18)
```

Nekonečně kouzelné jazykolamy

Kouzelná hůlka je až neuvěřitelně rychlá a přesná. I kdyby se čaroděj snažil sebevíc, kouzelnou větu (+ 123 456) vyhodnotí kouzelná hůlka rychleji a bez chyb. A co teprve (* 123 456)!

Navíc se čarodějové snaží psát kouzelné věty chytře - aby jich napsali málo, ale kouzelná hůlka aby toho udělala hodně. Takže učí kouzelnou hůlku, aby kouzelné věty vykonávala pořád dokola, nebo aby zpracovala velké množství informací najednou.

Kouzelné sloveso je kouzelné sloveso: rekurze

Čarodějové často potřebují vykonat nějakou kouzelnou větu znovu a znovu a znovu. Psát stejné kouzelné věty pod sebe je nudné a navíc není vždy jasné, kolik kouzelných vět bude nakonec potřeba. V takových situacích čarodějové naučí kouzelnou hůlku nové kouzelné sloveso, které použijí na to, aby kouzelnou hůlku ono nové kouzelné sloveso naučili.

Nekonečná podstatná jména: seznamy

Kouzelná podstatná jména ukrývají informace, které chce čaroděj získat. Většinou chce čaroděj získat více než jednu informaci, a tak potřebuje kouzelná podstatná jména spojit. Před první kouzelné podstatné jméno přidá druhé, před druhé přidá třetí, před třetí přidá čtvrté a tak dále a tak dále a tak dále…

Epilog nekonečně kouzelných jazykolamů

Poslední dvě kapitoly nastínily, jak čarodějové přistupují k nekonečnu. Nekonečno je vlastně opakování.

Opakování je důležité, protože pomáhá čarodějům popsat složitější problémy. Navíc je kouzelná hůlka v opakování vážně dobrá. Jen je potřeba dát pozor, aby čaroděj stál nohama na zemi a k nekonečnu přistupoval pragmaticky - pokud chce od kouzelné hůlky výsledek vykonání kouzelných vět dostat, musí zajistit, že opakování jednou skončí.

lých položek nákupního seznamu, zjistíme rekurzivně kouzelným slovesem seznam->čísla, které se právě snažíme naučit kouzelnou hůlku. Novou cenu položky a zbytek nového seznamu spojíme do nového seznamu kouzelným slovesem **cons**.

```
(define (seznam->čísla nákupní-seznam)
  (if (null? nákupní-seznam)
      (list)
      (cons (string-length (car nákupní-seznam))
            (seznam->čísla (cdr nákupní-seznam)))))
```

Kouzelná hůlka teď čarodějovi ráda poskytne informaci o cenách jednotlivých položek nákupního seznamu a na kouzelnou větu

```
(seznam->čísla (list "sýr" "rýže" "zelí" "čaj"))
```

odpoví: `'(3 4 4 3)`

Pro ověření, že nám vše funguje, jak má, poslouží kouzelná věta

```
(= (kolik-písmen-seznamu (list "sýr" "rýže" "zelí" "čaj"))
   (sečti-čísla-seznamu
     (seznam->čísla (list "sýr" "rýže" "zelí" "čaj"))))
```

na kterou má kouzelná hůlka odpovědět: `#t`

Je ale potřeba kouzelnou hůlku naučit sečti-čísla-seznamu:

> *Omlouváme se, tady je jediné místo v knize,*
> *kde nám došel inkoust.*

Shrnutí seznamů

Seznam je kouzelné podstatné jméno, které obsahuje další kouzelná slova.

Vypadá trochu jako kouzelná věta, ale nemá sloveso a začíná apostrofem a závorkou `'(`, za kterými následují kouzelná slova oddělená mezerami a nakonec zase závorka `)`.

Čarodějové rádi používají seznamy spolu s rekurzí, protože jim pomáhají vyjádřit problémy, které se opakují.

Kouzelné sloveso je kouzelné sloveso: rekurze

Čarodějové si velmi rádi usnadňují práci. Třeba taková chůze do schodů je únavná, a tak typického čaroděje nenapadne nic lepšího, než říci kouzelné hůlce, aby to udělala za něj.

Čarodějové dost často bydlí v různých poschodích a potřebují tedy vystoupat různý počet schodů. Navíc ale rádi sdílejí své umění s ostatními. Tohle dilema nakonec vyřešil jeden čaroděj tím, že vymyslel kouzelnou větu

```
(define (vyjdi-schod ze-schodu)
  (+ 1 ze-schodu))
```

kterou kouzelné hůlce předal několikrát za sebou, protože aby se dostal domů, musí vyjít tři schody:

```
(vyjdi-schod 0) (vyjdi-schod 1) (vyjdi-schod 2)
```

Výsledek, který je v DrRacket vidět v REPL, je:

```
1
2
3
```

Kouzelná věta která začíná kouzelným slovesem vyjdi-schod se tak hodí i čarodějovi, který potřebuje vystoupat pět schodů. Jednoduše kouzelné hůlce předá kouzelnou větu vícekrát:

```
(vyjdi-schod 0)
(vyjdi-schod 1)
(vyjdi-schod 2)
(vyjdi-schod 3)
(vyjdi-schod 4)
```

s výsledkem:

```
1
2
3
4
5
```

Když kouzelná hůlka vykoná kouzelnou větu začínající slovesem vyjdi-schod, výsledek je číslo schodu, na kterém se čaroděj ocitne. Když jde čaroděj domů, vždycky začíná v přízemí.

(define přízemí 0)

Když má čaroděj k dispozici kouzelné sloveso vyjdi-schod a kouzelné podstatné jméno přízemí, může nechat kouzelnou hůlku, aby jej dostala do třetího poschodí kouzelnými větami

```
(vyjdi-schod přízemí)
(vyjdi-schod 1)
(vyjdi-schod 2)
```

Pomocí kouzelného slovesa **let*** z kapitoly Pojmenování informací: definice proměnných může čaroděj místo 1 a 2 použít kouzelná podstatná jména první-schod a druhý-schod:

```
(let* ((první-schod (vyjdi-schod přízemí))
       (druhý-schod (vyjdi-schod první-schod)))
  (vyjdi-schod druhý-schod))
```

Výsledek je potom: 3

A protože výsledek vykonání kouzelné věty je nějaké slovo kouzelného jazyka, může čaroděj výsledné slovo použít v další kouzelné větě a vytvořit tak jednu dlouhou mocnou kouzelnou větu:

```
(vyjdi-schod (vyjdi-schod (vyjdi-schod přízemí)))
```

Výsledek je stejný, tedy: 3

Tenhle přístup ale rozhodně netěší úplně všechny čaroděje. A obzvláště netěší ty, co bydlí v mrakodrapu, a musí proto domů vystoupat 20 schodů:

```
(vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod
(vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod
(vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod
(vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod
přízemí))))))))))))))))))))
```


Procházení seznamu

Zjistit délku seznamu znamemá spočítat počet jeho položek. Prázdný seznam má 0 položek; pro každou položku seznamu se jeho délka zvětší o + 1.

Říká se, že kouzelná hůlka *projde* seznam a každou položku seznamu *použije*. Pro délku seznamu *použít* znamená přičíst + 1.

V kouzelnickém obchodě zaplatí čaroděj za každou položku tolik, kolik má položka písmen. Proto čaroděje dost zajímá, kolik písmen dohromady jeho nákupní seznam má. Pro každou položku seznamu tak místo + 1 použije čaroděj **string-length**:

```
(define (kolik-písmen-seznamu seznam)
  (if (null? seznam)
      0
      (+ (string-length (car seznam))
         (kolik-písmen-seznamu (cdr seznam)))))
```

Na kouzelnou větu

```
(kolik-písmen-seznamu (list "sýr" "rýže" "zelí" "čaj"))
```

pak kouzelná hůlka odpoví: 14

Vytváření seznamu

Nakonec by čaroděje mohlo ještě zajímat kolik jej stojí jednotlivé položky. A my bychom mohli projít nákupní seznam, z každé položky pomocí **string-length** zjistit cenu položky a všechny ceny postupně spojit do nového seznamu čísel. To by šlo.

Nejdřív je potřeba rekurzivně promyslet, jak kouzelné sloveso seznam->čísla kouzelnou hůlku naučit. Kdy procházení skončí? Když je nákupní-seznam prázdný. Když je nákupní-seznam prázdný, co je výsledek výkonání (seznam->čísla (**list**))? Nový prázdný seznam (**list**).

To byla *ukončující podmínka*. Teď *rekurzivní krok*.

Cenu položky z nákupního seznamu zjistíme kouzelným slovesem **string-length**. Zbytek nového seznamu, tedy zbývající ceny zby-

Délka seznamu

Čarodějové používají seznamy velmi často - líbí se jim více kouzelných slov v jednom podstatném jméně. Klasický kouzelnický učebnicový příklad práce se seznamem je zjištění délky seznamu:

```
(délka-seznamu (list "sýr" "rýže" "zelí" "čaj"))
```

Jenže kouzelná hůlka takové kouzelné sloveso nezná a odpoví:

```
délka-seznamu: undefined;
```

Velkou částí práce čaroděje je hledat, co se v problémech, které řeší, opakuje. Zkušenější čaroděj v problému délky seznamu opakování pozná hned: nejkratší možný seznam, prázdný seznam (**list**), má délku 0; vždy po přidání nějaké položky do seznamu se délka takového seznamu zvětší o 1.

Čaroděje takové rekurzivní přemýšlení přivede k nápadu, jak kouzelnou hůlku kouzelné sloveso délka-seznamu naučit: Když je seznam prázdný, jeho délka je 0. Pokud prázdný není, jeho délka je délka zbytku seznamu + 1:

```
(define (délka-seznamu seznam)
  (if (null? seznam)
      0
      (+ 1 (délka-seznamu (cdr seznam)))))
```

Když to pak čaroděj zkusí znovu a na délku nákupního seznamu se kouzelné hůlky zeptá stejnou kouzelnou větou

```
(délka-seznamu (list "sýr" "rýže" "zelí" "čaj"))
```

kouzelná hůlka už bude vědět a odpoví: 4

Délka seznamu je klasický kouzelnický učebnicový příklad práce se seznamem proto, že kouzelná hůlka kouzelné sloveso pro zjištění délky seznamu zná - je to **length** a čarodějové jej používají dost často. Proto na kouzelnou větu

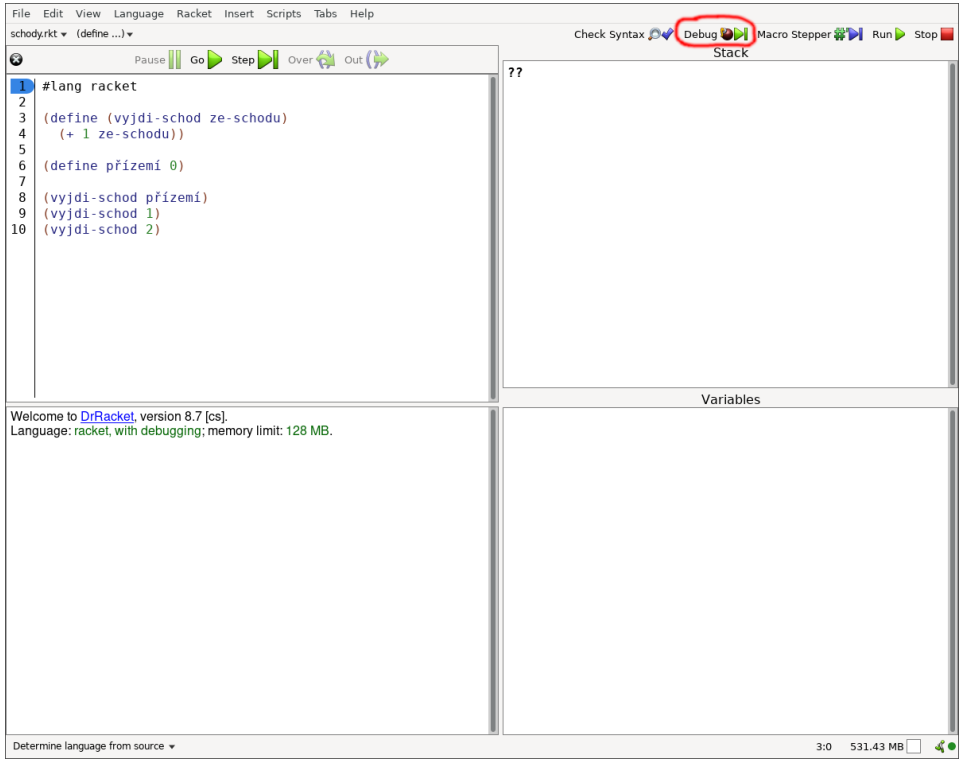
```
(= (délka-seznamu (list "sýr" "rýže" "zelí" "čaj"))
   (length (list "sýr" "rýže" "zelí" "čaj")))
```

odpoví kouzelná hůlka: #t

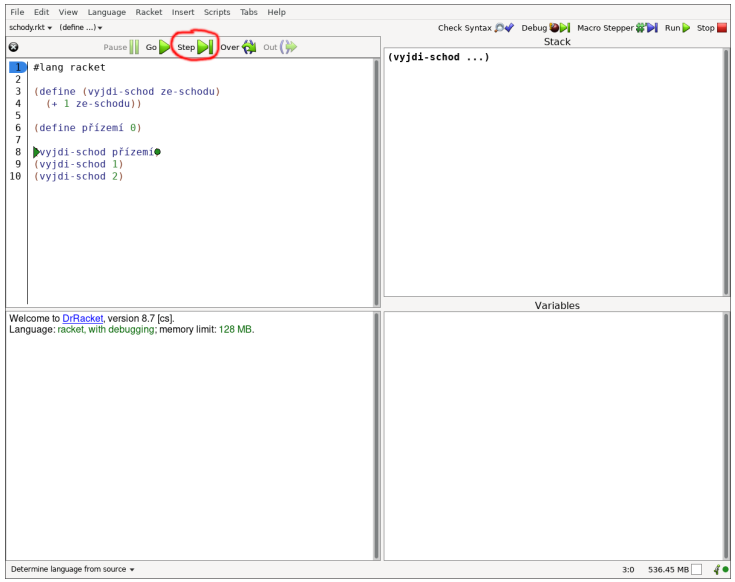
Vykonávání krok za krokem

Teď je nejspíš ten nejlepší čas na kouzelnický trik. Zkušenější čarodějové si dovedou představit, co asi tak kouzelná hůlka provede s kouzelnou větou, kterou čaroděj napíše nebo si někde přečte. A nebo taky ne.

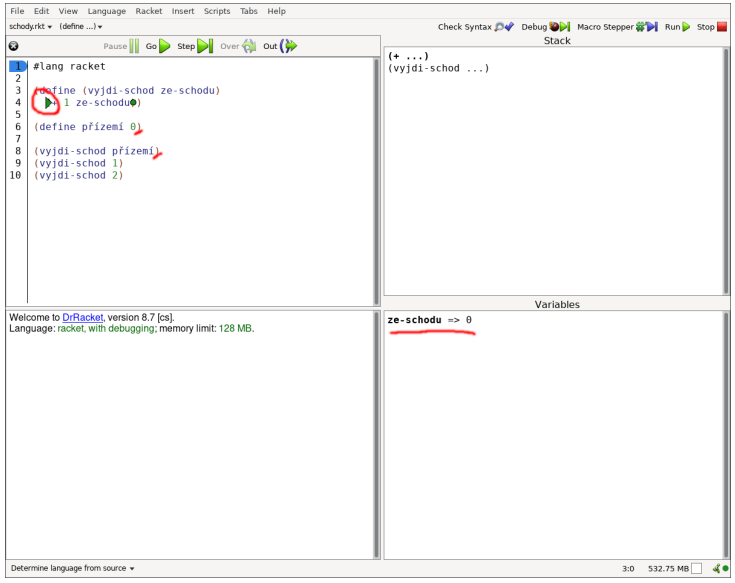
Když jsou čarodějové v koncích a kouzelná hůlka po vykonání kouzelné věty odpovídá úplně něco jiného, než čaroděj očekává, může čaroději v pochopení kouzelné věty pomoci vývojové prostředí (v našem případě DrRacket) a jeho debugger. Debugger čaroději pomůže projít dopis pro kouzelnou hůlku krok za krokem.



Obrázek 1: Zapnutí debuggeru ve vývojovém prostředí DrRacket.



Obrázek 2: Kliknutím na *Step* čaroděj *krokuje* vykonávání dopisu pro kouzelnou hůlku.



Obrázek 3: *Zelený trojúhelník* značí v jakém kroku vykonávání kouzelná hůlka je a *Variables* zobrazuje hodnotu kouzelný slov.

Seznam, který nemá žádnou položku, je prázdný seznam. Vytvoří se kouzelnou větou začínající kouzelným slovesem **list**, která ale nemá žádná další kouzelná slova:

```
(list)
```

Kouzelná hůlka pak odpoví: '()

Zda je seznam prázdný zjistí čaroděj kouzelnou větou začínající kouzelným slovesem **null?**. Na kouzelnou větu

```
(null? (list))
```

odpoví kouzelná hůlka: #t

Nakonec to chce prozkoumat **car** a **cdr** v souvislosti se seznamy různých délek.

Na (**car** '("zelí" "čaj")) odpoví kouzelná hůlka: "zelí"

Na (**cdr** '("zelí" "čaj")) řekne: '("čaj") protože i když zbytek seznamu obsahuje jenom jednu položku, je to pořád seznam.

Výsledek vykonání (**car** '("čaj")) je: "čaj"

A z kouzelné věty (**cdr** '("čaj")) udělá kouzelná hůlka: '()

Co s kouzelnou větou (**car** '())? Kouzelná hůlka řekne:

```
car: contract violation
  expected: pair?
  given: '()
```

Tedy: "Chyba! Prázdný seznam nemá první položku!"

Stejně tak má kouzelná hůlka výhrady vůči kouzelné větě (**cdr** '()), na kterou odpoví:

```
cdr: contract violation
  expected: pair?
  given: '()
```

Což je: "Chyba! Prázdný seznam nemá zbytek!"

Nekonečná podstatná jména: seznamy

Seznam je několik kouzelných slov dohromady. Čarodějové vytváří seznamy pomocí kouzelných vět začínajících kouzelným slovesem **list**. Jako třeba čarodějův nákupní seznam:

```
(list "sýr" "rýže" "zelí" "čaj")
```

Výsledkem vykonání takové kouzelné věty je kouzelné podstatné jméno

```
'("sýr" "rýže" "zelí" "čaj")
```

které začíná apostrofem ' a pak vypadá jako kouzelná věta se špatnou gramatikou, protože jí chybí sloveso.

Seznam je tedy kouzelné podstatné jméno, které začíná apostrofem a závorkou '(, pokračuje výčtem kouzelných slov oddělených mezerami a končí zase závorkou).

První položka a zbytek seznamu

Seznam seskládá z kouzelného slova, které je v seznamu první, a z dalšího seznamu, který je tvořen zbylými kouzelnými slovy.

První kouzelné slovo ze seznamu získáme pomocí **car**. Kouzelná hůlka na kouzelnou větu

```
(car (list "sýr" "rýže" "zelí" "čaj"))
```

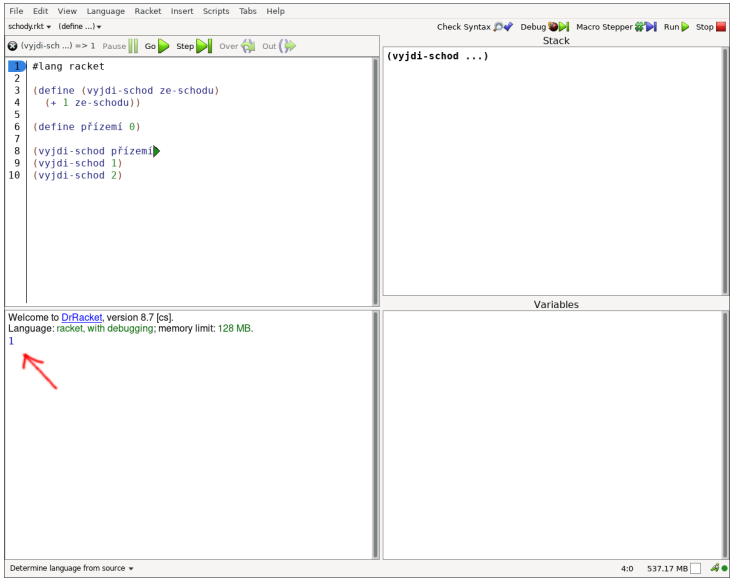
odpoví: "sýr"

Pro získání zbytku kouzelných slov slouží **cdr**, takže na

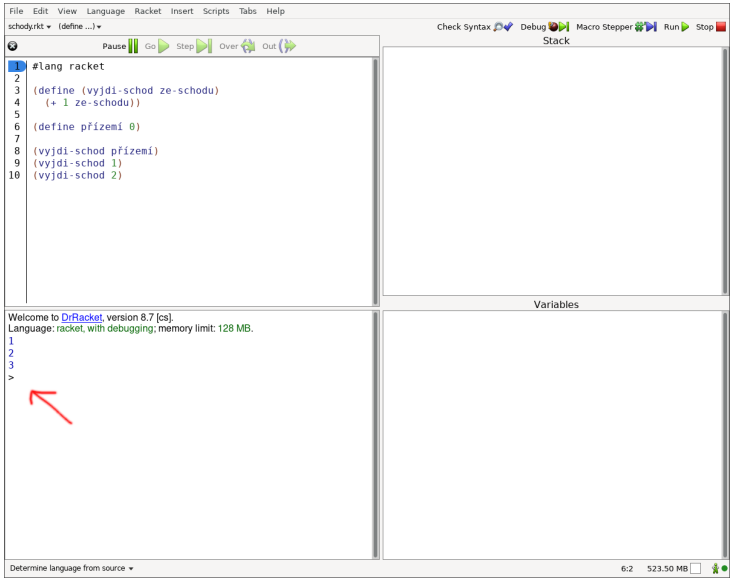
```
(cdr (list "sýr" "rýže" "zelí" "čaj"))
```

odpoví kouzelná hůlka: '("rýže" "zelí" "čaj")

Seznam je kouzelné podstatné jméno, takže je v pořádku, že je v kouzelné větě až po kouzelném slovesu **car** nebo **cdr**. Položky seznamu jsou ale kouzelná slova - mohou to být kouzelná podstatná jména, jako v nákupním seznamu čaroděje, ale i kouzelná slovesa.



Obrázek 4: Po vykonání kouzelné věty se výsledek kouzelné věty ukáže v okně výpisu.



Obrázek 5: Po proklikání všech kroků dopisu pro kouzelnou hůlku se v okně výpisu objeví "zobáček" > značící konec programu.

Obrázky (občas se říká *screenshoty*) vývojového prostředí DrRacket z krokování (občas se říká *ladění* nebo *debugování*) dopisu pro kouzelnou hůlku (občas se říká *zdrojového kódu*) zabíraly spoustu místa. Nerad dávám domácí úkoly, ale neprogramátor by si teď měl krok za krokem projít:

```
#lang racket
(define (vyjdi-schod ze-schodu)
  (+ 1 ze-schodu))

(define přízemí 0)

(let* ((první-schod (vyjdi-schod přízemí))
       (druhý-schod (vyjdi-schod první-schod)))
  (vyjdi-schod druhý-schod))
```

a určitě také:

```
#lang racket
(define (vyjdi-schod ze-schodu)
  (+ 1 ze-schodu))

(define přízemí 0)

(vyjdi-schod (vyjdi-schod (vyjdi-schod  přízemí)))
```

Krátká technická poznámka: První řádek dopisu pro kouzelnou hůlku, tedy #lang racket, je typický pro programovací jazyk *Racket*. Programovací jazyk *Racket* rozšiřuje jiný programovací jazyk, *Scheme*, který #lang ... nepoužívá.

Je zajímavé si při tom všimnou okna *Stack* (vpravo nahoře). Okno *Stack* říká, v jaké kouzelné větě se právě nachází kouzelná hůlka, která kouzelné věty vykonává. (V okně se zdrojovým kódem je aktuálně vykonávaná věta označena zeleným trojúhelníkem na začátku a zeleným kolečkem na konci. V okně *Stack* je vyznačena tučně.)

Shrnutí rekurze

Rekurze je, když čaroděj používá nové kouzelné sloveso ve chvíli, kdy kouzelnou hůlku ono nové kouzelné sloveso učí:

```
(define (počítej-do-dvaceti z-čísla)
  (if (= 20 z-čísla)
      z-čísla
      (počítej-do-dvaceti (+ 1 z-čísla))))
```

Na kouzelnou větu

```
(počítej-do-dvaceti 11)
```

pak kouzelná hůlka odpoví: 20

Rekurzivní funkce je počítej-do-dvaceti a *ukončující podmínka* je (= 20 z-čísla).

Kouzelná věta (počítej-do-dvaceti (+ 1 z-čísla)) je *rekurzivní krok*.

Rekurze se dá použít kdekoli je potřeba nějaké opakování a typickým příkladem může být kouzelná věta

```
(sečti-čísla-mezi 3 6)
```

na kterou má kouzelná hůlka odpovědět: 18

Kouzelné sloveso sečti-čísla-mezi naučí čaroděj kouzelnou hůlku kouzelnou větou:

```
(define (sečti-čísla-mezi menší větší) ; včetně
  (if (= menší větší)
      větší
      (+ menší (sečti-čísla-mezi (+ 1 menší) větší))))
```

Rekurzivní funkce sečti-čísla-mezi má *ukončující podmínku* (= menší větší) a *rekurzivní krok* je kouzelná věta:

```
(+ menší (sečti-čísla-mezi (+ 1 menší) větší))
```

Nebezpečná proto, že se může stát, že čaroděj bude stále stoupat vzhůru a nikdy se domů nedostane. Třeba když bydlí ve sklepě a číslo schodu je tak -1.

Proč se čaroděj bydlící ve sklepě nikdy nedostane domů? Protože čaroděj v kouzelném dopise říká kouzelné hůlce (stoupej-dokud-nejsem-doma přízemí). Kouzelné podstatné jméno přízemí je 0 a kouzelné sloveso vyjdi-schod způsobí, že čaroděj vystoupá o schod výše. Čaroděj proto nikdy neklesá a z přízemí 0 se tak nemůže dostat do sklepa -1.

To, aby kouzelná hůlka vykonávání rekurzivní kouzelné věty dokončila, zajišťuje *ukončující podmínka*. V případě kouzelné věty začínající kouzelným slovesem stoupej-dokud-nejsem-doma je ukončující podmínka jsem-doma?. V každém kroku kouzelná hůlka zkontroluje, jestli se čaroděj ze-schodu dostane domů. Pokud ano, odpoví kouzelná hůlka ze-schodu, tedy číslo schodu, na kterém čaroděj právě stojí. Jestliže ale kouzelná hůlka vykoná kouzelnou větu (jsem-doma? ze-schodu) a ona kouzelná věta neplatí - výsledek vykonání kouzelné věty je #f - kouzelná hůlka provede *rekurzivní krok*: znovu vykoná kouzelnou větu začínající kouzelným slovesem stoupej-dokud-nejsem-doma, tentokrát ale ze schodu (vyjdi-schod ze-schodu), tedy ze schodu o jeden výše.

V tuhle chvíli je dobrý nápad znovu v DrRacket spustit debugger.

Navíc je asi také ten správný čas zmínit *Indukci*, protože rekurze je paralela matematické indukce. *První krok* matematické indukce odpovídá tomu, když platí ukončující podmínka. *Indukční krok* pak zajišťuje pokračování ve vykonávání *rekurzivní funkce* v případě, že ukončující podmínka neplatí. Indukčnímu kroku se říká *rekurzivní krok* nebo *rekurzivní volání funkce*. Rekurzivní funkce je v tomto případě stoupej-dokud-nejsem-doma.

Třeba

```
(vyjdi-schod ...)
(let* ...)
```

říká, že kouzelná hůlka ve větě začínající kouzelným slovesem **let*** narazila na kouzelnou větu začínající kouzelným slovesem vyjdi-schod a nejprve musí vykonat kouzelnou větu začínající vyjdi-schod, aby mohla pokračovat ve vykonávání kouzelné věty začínající kouzelným slovesem **let***.

Podobně je to i v případě, když okno *Stack* zobrazí

```
(+ ...)
(vyjdi-schod ...)
(let* ...)
```

protože kouzelná věta, která začíná kouzelným slovesem vyjdi-schod, a zároveň je v tomto případě uvnitř kouzelné věty začínající **let***, obsahuje kouzelnou větu, která začíná kouzelným slovesem +.

Když čaroděj krokuje mocnou kouzelnou větu

```
(vyjdi-schod (vyjdi-schod (vyjdi-schod přízemí)))
```

zobrazí se v okně *Stack* dokonce čtyři řádky:

```
(+ ...)
(vyjdi-schod ...)
(vyjdi-schod ...)
(vyjdi-schod ...)
```

Stoupej, dokud nejsi doma

Jedním ze způsobů, jakým čaroděj vytvoří dlouhou mocnou kouzelnou větu je, že čaroděj do kouzelné věty *vnoří* další kouzelnou větu. Vnořená kouzelná věta jistě může začínat stejným kouzelným slovesem, jako třeba:

```
(vyjdi-schod (vyjdi-schod (vyjdi-schod přízemí)))
```

Čaroděj bydlící až na dvacátém schodu, musí vět vnořit více:

```
(vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod
(vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod
(vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod
(vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod (vyjdi-schod
přízemí))))))))))))))))))))
```

Když ale čaroděj, nebo i ne-čaroděj, stoupá po schodech domů, tak schody nepočítá. Že je doma, pozná ne-čaroděj podle vstupních dveří. Nebo rohožky. No a čaroděj se zeptá kouzelné hůlky, třeba kouzelnou větou začínající kouzelným slovesem jsem-doma?, na kterou odpoví kouzelná hůlka: #t nebo #f

Pro čaroděje bydlícím na třetím schodu tak ke kouzelnému slovesu vyjdi-schod přibude:

```
(define (jsem-doma? schod)
  (= 3 schod))
```

a pro čaroděje bydlícím na dvacátém schodu:

```
(define (jsem-doma? schod)
  (= 20 schod))
```

No a když čaroděj naučil kouzelnou hůlku kouzelným slovesům vyjdi-schod a jsem-doma?, chtěl by, aby kouzelná hůlka pochopila, když ji čaroděj řekne (stoupej-dokud-nejsem-doma přízemí).

Chtěl by, aby na takovou větu kouzelná hůlka odpověděla: 3

Nebo: 20

A tak kouzelnou hůlku naučí:

```
(define (stoupej-dokud-nejsem-doma ze-schodu)
  (if (jsem-doma? ze-schodu)
      ze-schodu ; jsem doma!
      (stoupej-dokud-nejsem-doma (vyjdi-schod ze-schodu))))
```

Krátká technická poznámka: Středník `;` značí *komentář*. Vše za ním, až do konce řádky, kouzelná hůlka ignoruje. Čarodějové tak sdílí poznámky, které jsou určené ostatním čarodějům a kouzelná hůlka o nich nemá vědět.

Celý dopis pro kouzelnou hůlku, který píše čaroděj bydlící ve třetím patře, tedy zní:

```
#lang racket

(define (vyjdi-schod ze-schodu)
  (+ 1 ze-schodu))

(define přízemí 0)

(define (jsem-doma? schod)
  (= 3 schod))

(define (stoupej-dokud-nejsem-doma ze-schodu)
  (if (jsem-doma? ze-schodu)
      ze-schodu ; jsem doma!
      (stoupej-dokud-nejsem-doma (vyjdi-schod ze-schodu))))
```

Když potom čaroděj kouzelné hůlce napíše

```
(stoupej-dokud-nejsem-doma přízemí)
```

kouzelná hůlka odpoví: 3

Čaroděj bydlící ve dvacátém patře napíše ten samý dopis, jen trochu změní jsem-doma?, protože bydlí výše.

V tuhle chvíli je dobrý nápad v DrRacket spustit debugger. Určitě není potřeba všechno hned pochopit. Jistě se ale vyplatí se teď trochu potrápit a sledovat, jaká kouzelná věta se při kterém kroku vykonává (zelený trojúhelník a kolečko a tučný řádek v okně *Stack*), jakou hodnotu právě mají kouzelná podstatná jména (proměnné, okno *Variables*) a jak to všechno dopadne (okno s výsledkem, kde je nakonec >).

Čaroděj tedy kouzelnou hůlku naučil stoupej-dokud-nejsem-doma. Aby kouzelnou hůlku nové kouzelné sloveso naučil, použil čaroděj kouzelná slovesa jsem-doma?, vyjdi-schod a … stoupej-dokud-nejsem-doma! Takovému přístupu se říká *rekurze*.

Rekurze je mocná a nebezpečná. Mocná proto, že pomáhá čarodějům bydlícím ve dvacátém patře se psaním dopisů pro kouzelnou hůlku.