Brainfuck - programování jako hlavolam

Jan Sedlák

17. června 2010

Obsah

1 Úvod

Pokud čtete tyto řádky, nejspíše jste se již rozhodli (nebo stále váháte, tudíž se Vás zde budu snažit rozhřešit) pro použití brainfucku v praxi, v projektu nebo či jen tak pro zábavu. Blahopřejeme, vybrali jste si ten nejsložitější a zároveň nejjednodušší, nejkomplexnější a zároveň nejprostší, nejkontroverznější a nejparadoxnější jazyk na celém svobodném Internetu. V dnešním komerčním prostředí se používá nejrůznějších jazyků (v dnešní době bohužel nejčastěji Java), další spousta jazyků není používaných v praxi, ale funguje dobře na akademické půdě. Stranou však stojí jazyk brainfuck. Brainfuck se, aspoň z počátku, zdá jako jazyk velmi neužitečný, a on jím věru doopravdy po všech stránkách je. Naproti tomu, pokud hledáte dobrodružství mezi programovacími jazyky (ať to může znít sebešíleněji) a toužíte pro "robinsonovskému" přístupu, je brainfuck přesně pro vás. Ze své vlastní zkušenosti mohu říct, že pocitu vítězství, který budete zažívat po napsání první if-then-else smyčky, se mnoho věcí nevyrovná. Jako takový proto musíte jazyk brát - jako hříčku, spíše jako hlavolam. Pokud ale přece jenom toužíte po určité praktičnosti, kontaktujte mě. Hodlám vyvinout brainfuckovského daemona, který bude běžet na pozadí a uchovávat data. Dokáži si představit jeho užití v jednodušších projektech namísto monstrózního SQL - pokud budete potřebovat uchovat čistě hodnoty, bude brainfuckovská metoda přístupu k datům nedocenitelná. V této knize se Vás budu snažit nejprve naučit základy brainfucku a pak Vám ukáži použití brainfucku v pythonovském projektu. Úvod bych zakončil citátem z Taa programování: "Na světě existuje spousta jazyků. Každý jeden z nich má ale své místo v srdci Taa.". Doufám, že i vy tu eleganci v brainfucku najdete a necháte se unášet jeho geniální jednoduchostí.

2 Metody přístupu, rezervovaná “slova”

Jazyk brainfuck má velice specifickou metodu přístupu k programování. Zatímco v každém jazyce se na začátku učíte tisíce datových typů, syntaxe a sémantiky, v brainfucku se neučíte nic. Zhola nic. Ale abychom Vás neošidili o druhou kapitolu, povíme si něco o stylu programování v brainfucku. V brainfucku se programuje pomocí operací prováděných nad nekonečným polem charů. Tak by zněla definice, řekněme to jednodušeji - na začátku programu máte strukturu, která by se dala popsat jako pole čísel. Všechna čísla jsou automaticky nastavena na nulu (takže máte de facto nekonečně nul) a maximální hodnota každého čísla je 255. Máte také takový “ukazatel”, který ukazuje na určitou buňku pole (její poloha či index jsou nepodstatné údaje). Nyní k rezervovaným slovům. Znakem ‘+’ zvýšíte číslo v aktuální buňce (kam aktuálně ukazuje ukazatel) o jedna. Znakem ‘-’ číslo o jedna snížíte. Znakem ‘>’ posunete ukazatel po poli o jednu pozici doprava, znakem ‘<’ posunete ukazatel o jednu pozici doleva. Znak ‘.’ vypíše na znak, jehož ordinální hodnota v ascii tabulce odpovídá číslu, uloženému v aktuální buňce (takže, pokud je v aktuální buňce číslo 64, tečka vytiskne zavináč - pokud nechápete o čem jsem teď psal, hledejte na wiki heslo ascii). Znak ‘,’ načte z klávesnici znak a jeho ordinální hodnotu v tabulce ascii uloží do aktuální buňky. Aby byl jazyk turing-kompletní, je třeba dodat ještě znaky ‘[’ a ‘]’, přičemž kód, uzavřený v těchto závorkách, se vykonává neustále do kola, dokud není v aktuální buňce hodnota 0 (jak touto konstrukcí nahradit if-then-else podmínky si ukážeme v další kapitole). Vše ostatní, co se vyskytne v brainfuckovském kódu, veškeré ostatní znaky jsou brány jako komentář a ignorovány. Tuto kapitolu zakončíme konečně nějakým kódem, pokuste se ho sami pochopit. Pokud Vám to ale nepůjde, nezoufejte, podrobně si ho rozebereme v příští kapitole.

```
Vytiskne na standardni vystup zavinac
++++++++[>++++++++<-]>.
```

3 Programování v brainfucku

V předchozí kapitole jsme si popsali jazyk jako takový a vyjmenovali vlastnosti rezervovaných slov, v této kapitole se podíváme na samotné programování. Začneme příkladem kódu, který vytiskne zavináč:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++.
```

Hodnotu v aktuální buňce postupně zvyšujeme z defaultní nuly o jedničku až na hodnotu 64, poté tečkou vytiskneme znak v aktuální buňce, odpovídající ordinální hodnotě 64 - zavináč.

Tento kód opravdu moc efektivní není - nesmíme se tedy bavit o efektivitě vykonávání kódu, to je v brainfucku vedlejší. Důraz musíme brát hlavně na jednoduchost zápisu kódu - tím se brainfuck liší od všech ostatních jazyků. Při výše uvedeném způsobu by totiž jenom kód výpisu helloworldu vypadal následovně:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++.>++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++.>+++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++..>+++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++.>+++++++++++++++++++++++++++++++++++++++++
+++++.>++++++++++++++++++++++++++++++++.>+++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++.>+++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++.>
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++.>+++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++.>+++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++.>++
++++++++++++++++++++++++++++++.
```

Postupně nastavíte každou buňku na požadovanou hodnotu (přičítáním jedničky), vytisknete a poté se přesunete na následující buňku.

Tento kód je, lidově řečeno, naprosto hovadský. Určitého velkého zlepšení dosáhnete, pokud po vytisknutí hodnoty v aktuální buňce přičtete/odečtete od hodnoty a vytisknete aktuální buňku s jinou hodnotou. Kód by pak mohl vypadat takhle:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++.++++++++++++++++++++++++++++.+++++++
..+++.>++++++++++++++++++++++++++++++++++++++++++++.---
---------.<++++++++.--------.+++.------.--------.>+.
```

což opravdu už dost šetří místo. My si však ukážeme ještě lepší řešení a to za pomocí poslední konstrukce, co jsme si v této kapitole ještě nepředváděli, cyklu. Bude to opět onen kód pro vytisknutí zavináče:

```
++++++++[>++++++++<-]>.
```

Tak tenhle kód už vypadá opravdu o mnoho lépe. Tuto konstrukci si musíme blíže vysvětlit. Nejprve hodnotu v aktuální buňce nastavíme na osm a poté zavoláme cyklus. Cyklus bude probíhat do chvíle, dokud nebude v aktuální buňce nulová hodnota. V každém projetí cyklu se přepneme do vedlejší buňky, přičteme osm, poté se přepneme zpět do původní buňky a odečteme jedna. Jistě jste si dobře spočítali, že po ukončení cyklu (ve chvíli, kdy už v původní buňce bude nulová hodnota postupným odečítáním jedničky v každém průchodu cyklu) v druhé buňce bude hodnota 8*8 (8 za každý průchod cyklu * 8 za počet opakování, který jsme určili původní buňkou) což je 64, potřebná hodnota. Po ukončení cyklu se poté akorát přepneme do druhé buňky a vypíšeme její hodnotu. Ač se to nezdá, konstrukce [] je velmi mocnou součástí jazyka brainfuck. S pomocí něho můžeme implementovat for cyklus (jak jsme si právě ukázali), while cyklus (jak je nasnadě, ukončovací podmínkou je vždy aktuální buňka != 0) a if-then-else podmínka, jak si ještě ukážeme. Tuto kapitolu zakončíme helloworldem, nyní už s cykly:

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.++++
+++..+++.>++.<<+++++++++++++++.>.+++.------.--------.
>+.>.
```

4 Některé užitečné brainfuckovské konstrukce

A to je vše přátelé! Ne vážně, právě jsme Vám odhalili veškeré charakteristiky brainfucku. Nyní umíte celý brainfuck. Ale protože víme, že vstup do nového jazyka může být někdy složitější, ukážeme Vám zde pár užitečných konstrukcí, které se v brainfucku používají.

```
[-]
```

Jednoduché, ale efektivní. Kód co vymaže aktuální buňku.

```
[>+<-]
```

Opět velmi jednoduché. Přesune obsah jedné buňky do druhé. Pokud chcete kopírovat se zachováním původní buňky, bude kód vypadat následovně - potřebujete třetí buňku:

```
[>+>+<<-]>>[<<+>>-]
```

Pokud chcete používat konstrukci if-then-else, známou z jiných jazyků, bude už kód vypadat trochu složitěji. Následující kód načte znak a pokud se jedná o 'a', vypíše na standardní výstup 'B'. Pokud se jedná o 'b', vytiskne znak 'A':

```
>>>>>+<<<<<,>+++++++++[<----------->-]<-------[[-]>>>>>
-<<<<<++++++++[>++++++++<-]>+.[-]<[-]]>>>>>[-++++++++[
>++++++++<-]>++.[-]]
```

Kód nejprve nastaví tzv. "přepínač", na páté buňce na hodnotu 1. Poté se vrátí zpět a načte hodnotu z klávesnice. Od načtené hodnoty odečte 97 (což je ordinální hodnota znaku 'a', znak 'b' má ordinální hodnotu 98) a poté na aktuální buňce spustí cyklus. To znamená že pokud byl načtený znak a, je v aktuální buňce 0 a cyklus se nespustí. V cyklu se nejprve přesuneme na pozici přepínače a nastavíme ho na 0. Poté vytiskneme znak 'A'. Po ukončení tohoto cyklu se přesuneme na pozici přepínače a spustíme zde cyklus vypsání 'B'. Pokud byl načtený znak a, nedošlo k přepnutí přepínače v minulém cyklu a proto se cyklus spustí a znak vytiskne. Pokud proběhl minulý cyklus, načtený znak byl b a přepínač byl přepnut. Proto se nespustí tento cyklus. Poznámka - jistě se ptáte, proč jsme nevypsali 'A' u 'a' a 'B' u 'b'. Je to jednoduše proto, že toho by se dalo dosáhnout i jednoduše odečtením konkrétní hodnoty a vytisknutím. Zatímco opačného efektu dosáhnete pouze výše uvedeným kódem.

5 Implementace mindfuck a použití pyfuku v Pythonovském kódě

Program mindfuck je pythonovská implemetace brainfuckovského interpreteru. Mindfuck je uvolněn pod licencí GNU/GPL (takže může každý nahlédnout do jeho zdrojových kódů) a má dohromady ani ne 200 řádků kódu. Mindfuck je testovaný aby plně podporoval standard jazyka brainfuck (mohu osobně říci, že není moc jiných jazyků než Python, co by poskytovaly tak efektivní konstrukce pro implemetaci brainfuckovského nekonečně velikého zásobníku). Mindfuck se ovládá velice jednoduše - ve svém oblíbeném textovém editoru napíšete brainfuckovský kód, soubor s kódem uložíte a poté pouze spustíte mindfuck a jako parametr mindfucku předáte cestu k souboru s kódem. Mindfuck pro interpretaci brainfucku používá pythonovský modul pyfuk, o kterém bychom Vám rádi pověděli více. Modul pyfuk načtete ve svém pythonovském programu jednoduše - umístíte soubor pyfuk.py do složky s programem a v kódu napíšete "import pyfuk". Modul pyfuk definuje jednu novou výjimku InterpretationError a hlavně třídu BrainInterpreter. V konstruktoru BrainInterpreter očekává předání argumentů writeFunction a debug. writeFunction je funkce pro standardní výstup programu - defaultní hodnota argumentu je sys.stdout.write. Pokud jste spokojení se vstupen z konzole, můžete tento argument vynechat. Další argument, debug, určuje, zda-li chcete s každým krokem interpreteru podrobné info o zásobníku, aktuální pozici v něm či aktuálně prováděném kódu. BrainInterpreter zavoláte v pythonu následovně:

```
...
import pyfuk
...
interpreter = pyfuk.BrainInterpreter()
...
```

BrainInterpreter má veřejně přístupnout instanční proměnnou BrainInterpreter.brainstack, což je seznam se zásobníkem brainfucku. Dále má třídní proměnnou reservedWords, což je seznam rezervovaných slov brainfucku. Nejdůležitější metodou BrainInterpreteru je metoda BrainInterpret.interpret(code). Jako argument očekává string s kódem. Metoda interpret provádí samotné interpretování kódu, používá k tomu soukromou metodu __interpretOneChar(char). Dále se musíme ještě zmínit o metodě BrainInterpreter.printf(), která slouží pro pohodlné vypsání zásobníku brainfucku. Použití pyfuku v Pythonu může vypadat následovně:

```
...
interpreter.interpret(",")
...
interpreter.interpret(".")
```

```
...
interpreter.brainstack[1] = 32
...
interpreter.interpret("[-]++++++++[>++++++++<-]>.")
...
```

6 Závěr

Brainfuck nesmíte brát jako plnohodnotný jazyk. Nebude mít nikdy nejefektivnější dobu přístupu, rozhodně v něm jednoduše nenapíšete rozhraní programu a používat ho pro matematické výpočty se rovná profesní sebevraždě. Pokud ale k brainfucku přistupujete jako k zábavě a hlavolamu, jistě vás příjemně překvapí, jaké zázraky se dají vykouzlit s tak málo omáčkou. V porovnání s např. Javou jistě bezmezný ráj.