Handbók fyrir forritunarmálið Tush

Eiríkur Ingi Magnússon

16. apríl 2015

Útdráttur

Tush málið er frekar einfalt forritunarmál sem tekur sér forritunarmálin ruby og javascript til fyrirmyndar hvað málfræði varðar. Tush er dínamískt (dynamic), halaendurkvæmt forritunarmál sem styður einungis staðværar breytur og er bálkmótað hvað föll varðar. Tush keyrir á Morpho sýndarvélinni.

Efnisyfirlit

1 Inngangur

Tush málið er forritunarmál sem þýðist yfir í Morpho smalamál. Tush þýðandinn er skrifaður í `javascript` og hægt er að nálgast þýðandann ásamt keyrsluhæfu umhverfi á `https://github.com/Eiiki/tush`.

Hægt er að prófa Tush af vefnum í gegnum slóðina `https://notendur.hi.is/~eim5/TOL202M/`. En til þess að keyra og fá bitastæða útkomu út úr forritstexta sem skrifaður er í málinu að þá þarf `Node.js`[1] að vera uppsett á viðkomandi vél ásamt ásamt `Morpho`[2] eða `Morpho.jar` skráin þarf að vera til staðar.

Þar sem að Tush keyrir á Morpho sýndarvélinni að þá getur það nýtt sér öll þau innbyggðu föll sem finna má í `BASIS`[3] einingunni úr Morpho.

[1] https://nodejs.org/
[2] http://morpho.cs.hi.is/
[3] http://morpho.cs.hi.is/docs/Morpho.pdf#page=71

Dæmi um einfaldan halaendurkvæman forritsbút úr Tush:

```
#Use:    x = fibo(n) or x = fibo()
#Before: n is an number ≥ 0 or nothing
#After:  x is the n'th fibonacci number if n is
#        presented, else it is the 10th
#        fibonacci number
def fibo(n=10)
    if(n <= 0)
        return 0;
    elsif( n == 1)
        return 1;
    end
    return fibo(n-1) + fibo(n-2);
end
```

2 Notkun og uppsetning

Til þess að geta keyrt og þýtt forrit skrifuð í Tush er ráðlagt að sækja keyrslumöppuna af `https://github.com/Eiiki/tush`.

Í keyrslumöppunni er að finna skránna `code.tsh` og inniheldur hún löglegt mál skrifað í Tush. Hana er hægt að þýða með skipuninni

```
node tush.js code.tsh
```

En þá verður til ný skrá, `code.mexe`, sem inniheldur jafngild Morpho smalamál og því sem var í `code.tsh` fyrir þýðingu.

Smalamálið má svo þýða með Morpho sýndarvélinni með skipunninni

```
java -jar morpho.jar -c code.mexe
```

Svo er hægt að keyra hana að lokum með skipuninni

```
java -jar morpho.jar code
```

3 Málfræði

Málfræðin í Tush reynir að vera skýr og læsileg hvað forritunarmál varðar. Bil og þar af leiðandi línubil skipta engu máli þar sem þau eru hunsuð af þýðandanum.

3.1 Frumeiningar málsins

3.1.1 Athugasemdir

Athugasemdir byrja á tákninu #. Allt það sem kemur á eftir tákninu # í viðkomandi línu er skilgreint sem athugasaemd og mun þýðandinn hunsa þau á keyrslutíma.

3.1.2 Lykilorð

Lykilorð í Tush eru sem hér segir:
`"var", "def", "while", "if", "elsif", "else", "end", "return", "true", "false", "null", "not", "or", "and".`

Einnig eru eftirfarandi tákn frátekin í málinu:
`";", "!=", "!", "||", "&&", "<=", ">=", "<", ">", "==", "++", "+", "-", "*", "/", "%", "=", "(", ")", "[", "]", ","`

3.2 Málrit

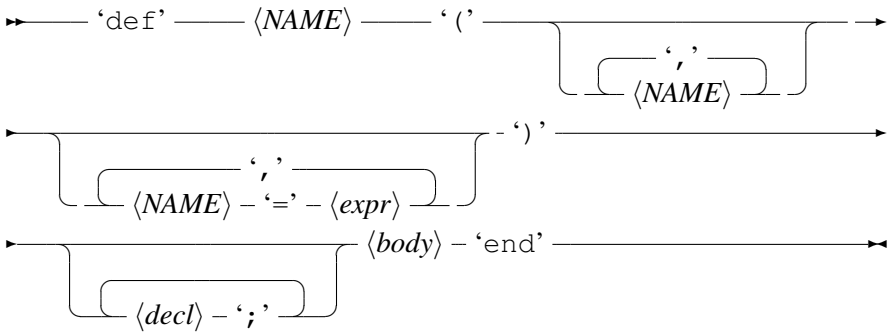
Hér að neðan má sjá málrit sem lýsir Tush málinu.

3.2.1 Forrit

⟨*program*⟩:

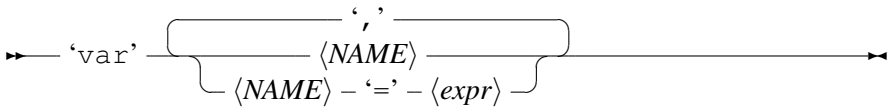
3.2.2 Föll

⟨*function*⟩:



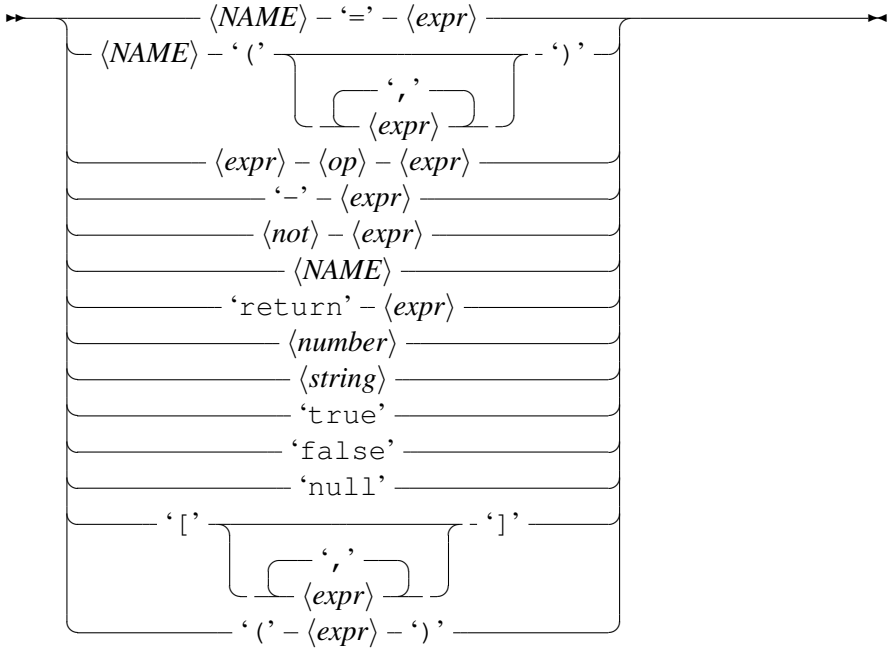
3.2.3 Skilgreiningar

⟨*decl*⟩:

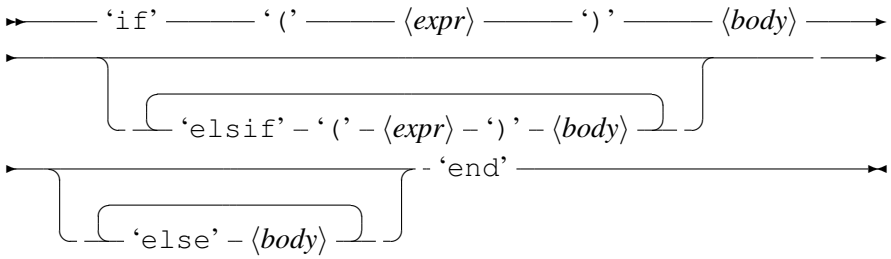


3.2.4 Segðir

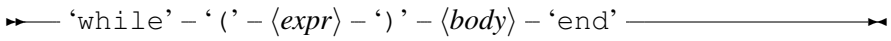
⟨*expr*⟩:



⟨*if_expr*⟩:



⟨*while_expr*⟩:

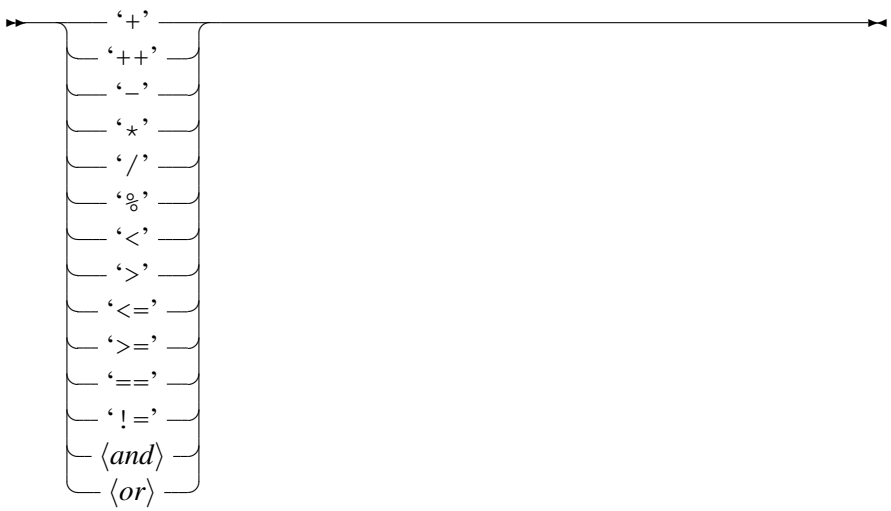


3.2.5 Stofnar

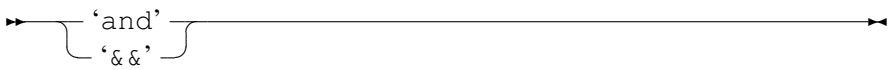
⟨*body*⟩:

3.2.6 Aðgerðir

⟨*op*⟩:



⟨*and*⟩:



⟨*or*⟩:

⟨*not*⟩:

4 Merking málsins

4.1 Gildi

Gildi í Tush geta verið heiltölur, fleytitölur, strengir, true, false eða null.
Í Tush eru gildin `false` og `null` ósanngild, öll önnur gildi eru skilgreind sem sönn.

4.2 Breytur

Breytur í Tush geta tekið gildin sem talin eru upp hér að ofan. Einnig geta breytur verið á formi lista og fylkja. En til að gera breytu að fylki þarf að nýta sér innbyggðu föllin úr `BASIS`[4] einingunni í Morpho.

Nýjar breytur eru síðan skilgreindar á eftirfarandi hátt:

```
1 var x;
2 x = <expr>;
```

eða:

```
1 var x = <expr>;
```

Þar sem `<expr>` er lögleg segð úr Tush sem lýst er með málriti í kafla 3.2.4.

[4] http://morpho.cs.hi.is/docs/Morpho.pdf#page=71

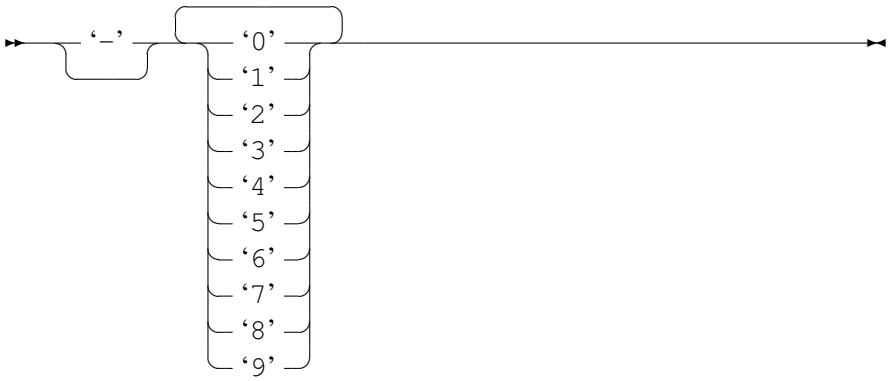
4.3 Merking segða

4.3.1 Heiltölusegð

Heiltölusegð í Tush er hagað eins og heiltölusegð í Morpho, en hún er einhver tala n, þar sem $-2,147,483,648 \leq n \leq 2,147,483,647$.

Málrit fyrir heiltölusegð er

⟨*integer*⟩:



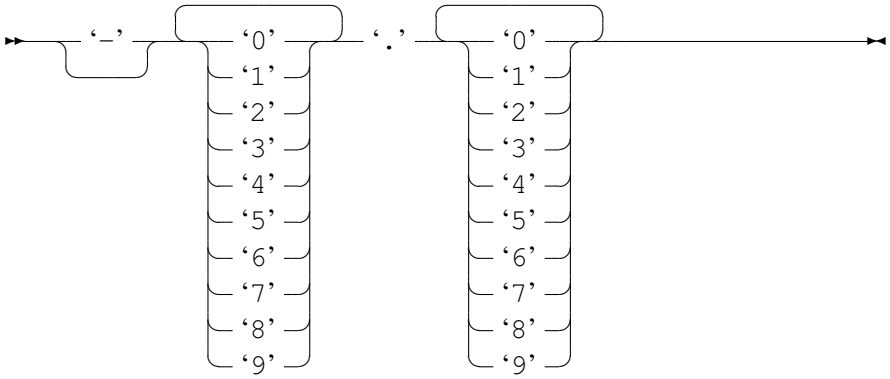
Athugum þó að yfirflæði getur átt sér stað ef við skilgreinum tölu stærri en $2,147,483,647$ eða minni en $-2,147,483,648$.

4.3.2 Fleytitölusegð

Fleytitölusegð í Tush er einhver tala n, þar sem $-\infty < n < \infty$.

Málrit fyrir fleytitölusegð er

⟨*double*⟩:



4.3.3 Strengsegð

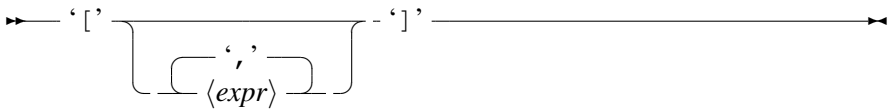
Strengsegð í Tush er hagað eins og strengsegð í Morpho, en hún er afmörkuð með tákninu ".

4.3.4 Listasegð

Listasegð í Tush er hagað eins og listasegð í Morpho, en hún einhver runa tákna sem afmarkast af táknunum '[' og ']', þar sem að hvert gildi í listanum er aðgreint með tákninu ','.

Málrit fyrir listasegð er

⟨*list*⟩:



4.3.5 return-segð

Return-segð í Tush er segð á forminu `return <expr>`. Slík segð getur aðeins komið fyrir innan stofns sérhvers falls. Þegar segðin kemur fyrir í keyrslu er hægri hlið hennar (<expr>), reiknuð út og sett sem skilagildi fallsins.

Athugum þá að í Tush að þegar við skilum gildi úr falli að þá hættum við strax í því falli og næsta fall tekur við og skilar sínu gildi til þess sem kallaði á upphaflega fallið. Við þetta að þá stækkar gildahlaðinn ekkert við djúpa endurkvæmni og étur þar af leiðandi ekki upp minni. En þessi hegðun er vegna þess að Tush er halaendurkvæmt forritunarmál.

4.3.6 Röksegðir

Röksegðir í Tush er hagað eins og röksegðum í Morpho.

Samaburðarvirkjarnir eru táknin <, >, <=, >= og ==. En þær eru í raun bara kall á samsvarandi Morpho fall, sem tekur inn tvær segðir og beytir tilheyrandi samanburðarvirkja á þær.

Rökvirkjarnir eru `and` (eða &&), `or` (eða ||) og `not` (eða !). En þeir skammhleypa þegar vinstri hlið segðar (þá <expr> 'and' <expr> eða <expr> 'or' <expr>) ákvarðar gildi heildar segðarinnar, þ.e. þeir reikna þá ekki út úr hægri hliðinni. Fyrir `and` segð þyrfti vinstri hlið segðarinn að vera ósönn til þess að skammhlaup

eigi sér stað og sönn fyrir `or` segð. Ósanngildi í Tush eru þau sömu og í Morpho, en þau eru `false` og `null`, annað er skilgreint sem sanngilt.

4.3.7 Kallsegð

Kallsegðir í Tush eru segðir á forminu '`fun(a`$_1$`, ..., a`$_n$`)`', þar sem $a_1 \ldots a_n$ eru 0 eða n breytur sem sendar eru sem inntök inn í fallið `fun`.

En það er hægt að skilgreina föll á mismunandi vegu í Tush, en sú hugmynd er fengin úr forritunarmálinu `Ruby`[5]. Tökum nokkur dæmi.

1.
```
#Use:    fun1(a,b) or fun1(a)
#Before: a and/or b are strings
#After:  The strings a and b have been joined
#        and printed to standard output
def fun1(str, str2="optional")
        println(str++str2);
end
```

Fallið `fun1` er hægt að kalla á tvennskonar vegu, en kallsegðin `fun1("optional parameters are ")` myndi prenta út fyrir okkur segðina `optional parameters are optional`. Hinsvegar, ef við myndum kalla á fallið með segðinni `fun1("optional parameters can be ","overwritten")` myndi prenta út fyrir okkur segðina `optional parameters can be overwritten`. Öll önnur köll á fallið, þ.e.a.s. fjöldi breyta sem sendar eru sem inntak í fallið, eru ólögleg köll.

2.
```
#Use:    fun2(a,b)
#Before: a and b are numbers
#After:  The sum of a and b has been printed
#        to standard output
def fun2(a,b)
        println(a+b);
end
```

Fallið `fun2` er einungis hægt að kalla á með kallsegð á forminu `fun2(a,b)` þar sem a og b eru einhverjar tölur.

[5]https://www.ruby-lang.org

3.
```
#Use:    fun3(a,b) or fun3(a) or fun3()
#Before: a and/or b are numbers
#After:  The sum of a and b, or a and 2, or 1 and 2
#        has been printed to standard output
def fun3(a=1, b=2)
        println(a+b);
end
```

Fallið `fun3` er hægt að kalla á, á þrennskonar vegu. Kallsegðin `fun3()` myndi prenta út fyrir okkur summu talnanna 1 og 2, kallsegðin `fun3(a)`, þar sem a er einhver tala, myndi prenta út fyrir okkur summu talnanna a og 2 og kallsegðin `fun3(a,b)`, þar sem a og b eru einhverjar tölur, myndi prenta út fyrir okkur summu talnanna a og b.

4.3.8 Tvíundaraðgerðir

Tvíundaraðgerðum í Tush er hagað eins og í Morpho, og eru þær í raun bara kall á samsvarandi Morpho fall, en þær eru '+', '-', '*', '/', '%', '++'.

4.3.9 Einundaraðgerðir

Einungis tvær einundaraðgerðir eru í Tush, en þær eru '$-$' og `not` virkinn (eða !). En einundarvirkinn '$-$' er kall á samsvarandi virkja í Morpho.

4.3.10 if-segð

If segð í Tush er lýst með málriti í kafla 3.2.4. En segðin er á forminu
`if(expr`$_1$`)` `body`$_1$ `elsif(expr`$_2$`)` `body`$_2$ `...` `elsif(expr`$_{n-1}$`)` `body`$_{n-1}$ `else` `body`$_n$ `end`
þar sem að n getur verið hvaða heiltala sem er ≥ 1.
Ef að `expr`$_i$ er sanngild að þá keyrist `body`$_i$ og farið út úr `if` segðinni, ef hún er hinsvegar ósönn að þá er athugað skilyrðið fyrir `expr`$_{i+1}$ og svoleiðis haldið áfram þar til að komið er að `else` klausunni og þá `body`$_n$ keyrð.
Málrit fyrir `body` má sjá í kafla 3.2.5.

4.3.11 while-segð

While segð í Tush er lýst með málriti í kafla 3.2.4. En segðin er á formin
`while(expr) body end`
Ef að `expr` segðin er ósönn að þá er `body` ekki keyrt og farið út úr `while` lykkjunni, ef hún er hinsvegar sönn að þá er `body` keyrt og að þeirri keyrslu lokið

er athugað hvort að `expr` segðin sé ennþá sönn, ef svo er að þá er `body` aftur keyrð o.s.frv.