Programmeerimiskeel APL

Raivo Laanemets

17. mai 2009. a.

Sissejuhatus

- APL - **A** **P**rogramming **L**anguage
- Kenneth E. Iverson (1920-2004)
  - Elukutselt matemaatik
  - Uuris matemaatilist notatsiooni
  - 1960 - läks tööle IBM-i
  - IBM-s kohtus Adin Falkoffiga, kes huvitus tema notatsiooni kasutamisest algoritmide kirjeldamisel
- 1964 - IBM/360 arhitektuuri kirjeldamiseks
- 1965 - APL keele esimene realisatsioon
- 1989 - APL standard (süntaks ja semantika) - ISO8485
- 1992 - A+ (*http://www.aplusdev.org*) APL realisatsioon UNIX-tüüpi süsteemidele (tööjaamadele)

Sissejuhatus

- APL realisatsioonid on interpretaatori kujul
- Programmeerija ning kasutaja töötavad programmiga interaktiivselt
  - Interpretaatorile antakse APL avaldis, mille tulemusena väärtustatakse muutuja või/ja väljastatakse midagi ekraanile
- APL avaldised kasutavad notatsiooni, mis sisaldab rohkelt mitte-ascii tähemärke
  - ⍳, ∈, ⍴, ×, ↓, ←, ⊛ jpt.
  - Programmiga töötamiseks vaja eriklaviatuuri
- Unikaalne keel - ei ole võrreldav teiste vähegi levinud keeltega

Sissejuhatus

Näide:

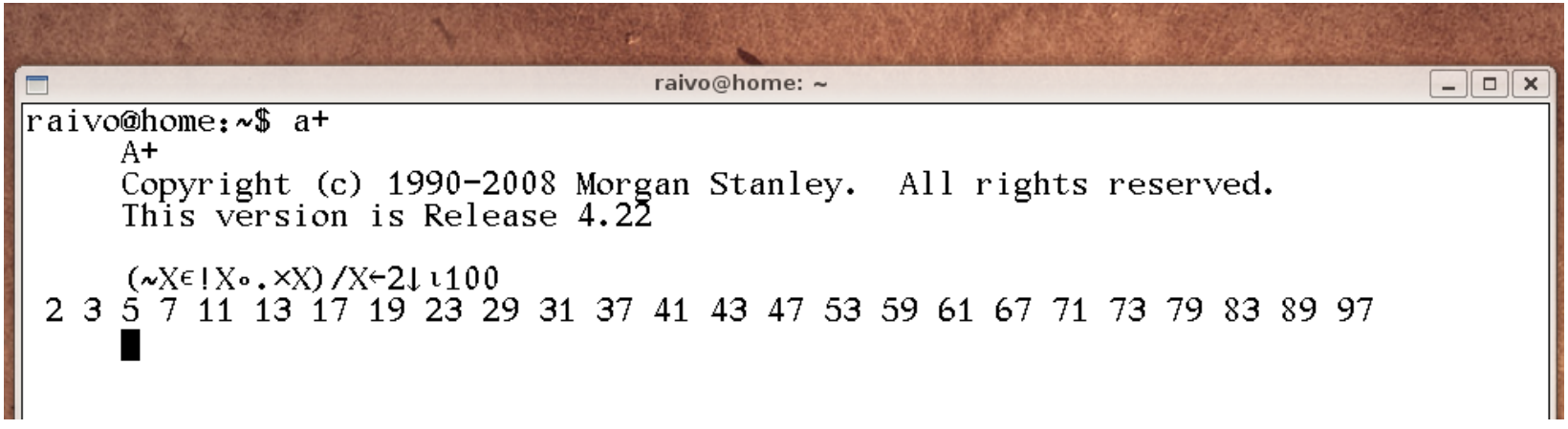
```
(~X∈!X∘.×X)/X←2↓⍳N
```

How to Shoot Yourself In the Foot:

"You shoot yourself in the foot; then spend all day figuring out how to do it in fewer characters."

Sissejuhatus

Näide:



APL konstruktsioonid

- Süntaks baseerub Iversoni matemaatilisel notatsioonil
  - "A Programming Language", Kenneth E. Iverson, 1962
  - "Notation as a Tool of Thought", Kenneth E. Iverson, 1980
- Primitiivsed lekseemid sarnased teistele keeltele (tähemärgid, stringid, arvud)
- APL süntaks formaliseeritav BNF formalismis (vähemalt osaliselt)
  - "The Syntax of APL, an Old Approach Revisited", Jean J. Girardot, Florence Rollin

APL konstruktsioonid

- Andmeühikuks on massiivid
- Andmetele rakendatakse funktsioone
- Funktsioonidele rakendatakse operaatoreid
- Tavalised aritmeetilised operaatorid on APL kontekstis funktsioonid!
- Avaldisi loetakse vasakult paremale: `3×4+5` väärtus on `27`
- Muutujad käituvad sarnaselt imperatiivsete keelte muutujatele
  - Väärtuse omistamise: `x←3×4+5`
  - Ei vaja deklareerimist
  - Omistamise väärtus on omistatava avaldise väärtus

Massiivid

- Kõik APL väärtused on massiivid (sh. numbrid, tähemärgid, funktsioonid)
- Massiivid võivad olla mitmemõõtmelised
- Mitmemõõtmelist massiivi iseloomustab tema kuju (*shape*)
  - Kuju esitab massiivi mõõtmete vektor (2×3 tabeli korral `2 3`)
- Ühemõõtmelised masiivid esitavad vektoreid, liste ja stringe
- Masiivi elementideks võivad olla teised massiivid (*nested array*)
- Sama massiiv võib sisaldada erinevat tüüpi andmeid (täisarvud, ujupunktarvud, tähemärgid)
- Massiivi elemente/alamosi saab kasutada indeksavaldiste abil

Massiivid

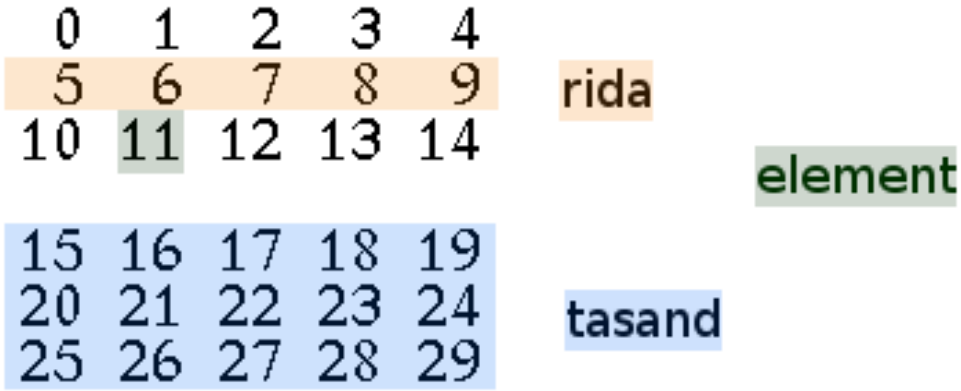
Massiivide esitamine:

- 3-elemendiline ühemõõtmeline massiiv: `10 2.3 34`
- 5-elemendiline sõnede massiiv: `` `sym1`sym2`sym3`sym4`sym5 ``
- 6-elemendiline tähemärkide massiiv (string): `'axrTVw'`
- Nestitud 6-elemendiline massiiv: `(`sym1;+;1 2 3 4;1.7 3.14;'example';)`
- Kahemõõtmeline massiiv: `2 3⍴'axrTVw'` - moodustatud kaheargumendilise funktsiooni ⍴ (rho) abil (vektor `2 3` määrab tulemuse kuju)

Massiivid

Massiivi genereerimine (intervallfunktsioon) `⍳` (iota):

- Vektor numbritest $0, 1, \ldots, 9$: `⍳10`
- Kolmemõõtmeline massiiv: `⍳2 3 5`



  - 2 tasandit
  - igal tasandil 3 rida
  - igal real 5 elementi

Massiivid

Massiivi elementide indekseerimine:

- Ühe elemendi selekteerimine: `'axrTVw'[4]`
- Mitme elemendi selekteerimine: `'axrTVw'[5 0 1]`
- Mitmemõõtmelisest massiivist selekteerimine: `(⍳2 3 5)[0;1;3]`
- Veeru selekteerimine: `(⍳2 3 5)[0;;4]` - väljastab `4 9 14`

```
 0  1  2  3  4
 5  6  7  8  9
10 11 12 13 14

15 16 17 18 19
20 21 22 23 24
25 26 27 28 29
```

Funktsioonid

- APL funktsioonid tavaliselt ühe või kahe argumendiga
- Ühe argumendiga f.-ni rakendamine: argument funktsionaalsümbolist vahetult paremal:

$$f\ x$$

  - f - funktsioonisümbol, x - argument

- Kahe argumendiga f.-ni rakendamine: üks argument vasakul, teine paremal (infiks-kuju):

$$y\ f\ x$$

  - f - funktsioonisümbol, x - esimene argument, y - teine argument

Funktsioonid

Kuna avaldisi loetakse paremalt vasakule, siis:

- $f_1\ f_2\ x \equiv f_1\ (f_2\ x)$ - f_1 ja f_2 üheargumendilised
- $f_1\ y\ f_2\ x \equiv f_1\ (y\ f_2\ x)$ - f_1 üheargumendiline, f_2 - kaheargumendiline
- $y\ f_1\ f_2\ x \equiv y\ f_1\ (f_2\ x)$ - f_1 kaheargumendiline, f_2 - üheargumendiline
- $y\ f_1\ x\ f_2\ z \equiv y\ f_1\ (x\ f_2\ z)$ - f_1 ja f_2 kaheargumendilised

Annab lihtsa viisi avaldiste parsimiseks ja interpreteerimiseks, töödeldes avaldist vasakult paremale

Funktsioonid

Näiteid ühe argumendiga aritmeetilistest funktsioonidest:

$\lvert$	absoluutväärtus	$-$	vastandarv
$\lceil$	ülemine täisosa	$\sim$	eitus
$\lfloor$	alumine täisosa	$\circ$	korrutamine π-ga
$\star$	eksponent	$\div$	pöördarv
$+$	samasusfunktsioon	$?$	juhuarvu genereerimine
$\circledast$	naturaallogaritm	$\times$	signum

Näited:

- π^2 alumise täisosa arvutamine: `⌊○○1`

Funktsioonid

Näiteid kahe argumendiga aritmeetilistest funktsioonidest:

$+$	liitmine	$\lfloor$	miinimum
$\wedge$	konjuktsioon	$\times$	korrutamine
$\circ$	trigonomeetriline f.	$\vee$	disjunktsioon
$\div$	jagamine	$*$	astendamine
$=,\geq,\leq,>,<,\neq$	võrdlemine	$\mid$	jääk
$\circledast$	logaritm	$-$	lahutamine
$\lceil$	maksimum		

Trigonomeetrilise funktsiooni vasakpoolseks argumendiks on soovitud konkreetne trigonomeetrilise funktsiooni nimi või sellele vastav number. Näiteks `2001` arvutab $\cos \pi$ väärtuse.

Funktsioonid

Funktsioone saab rakendada ka massiividele:

- Kahe vektori elemendikaupa liitmine: `1 2+3 4`
- Vektori elementide absoluutväärtuste vektor: `|3 ¯4`
- Vektori elementide läbikorrutamine 2-ga: `2×3 4`

Väljund interpretaatorist:

```
      1 2+3 4
4 6
      |3 ¯4
3 4
      2×3 4
6 8
```

Funktsioonid

Näiteid mittearitmeetilistest funktsioonidest:

- Vektorite konkateneerimine: `(⍳3),3 4`
- Elementide valik: `1 1 #3 4 5`
- Juhusliku 3-elemendilise vektori genereerimine: `3?10`
- 2 esimese elemendi eemaldamine vektorist: `2↓⍳10`

Väljund interpretaatorist:

```
      (⍳3),3 4
0 1 2 3 4
      1 1 #3 4 5
4 4
      3?10
3 8 0
      2↓⍳10
2 3 4 5 6 7 8 9
```

Operaatorid

- Operaatoreid kasutatakse funktsioonide käitumise laiendamiseks
  - Näiteks summafunktsiooni abil kõigi masiivi elementide summa leidmiseks
- Operaatorid jagunevad
  - Ühe funktsiooni-argumendiga (*monadic*)
  - Kahe funktsiooni-argumendiga (*dyadic*)

Näiteid:

- Elementide summa leidmine (*reduce*-operaator /): `+/⍳100`

Operaatorid

Massiivide otsekorrutis (*outer product* `∘.f`):

- `f` - kahekohaline funktsioon

Näide: 10×10 korrutustabel

```
      V∘.×V←⍳10
 0  0  0  0  0  0  0  0  0  0
 0  1  2  3  4  5  6  7  8  9
 0  2  4  6  8 10 12 14 16 18
 0  3  6  9 12 15 18 21 24 27
 0  4  8 12 16 20 24 28 32 36
 0  5 10 15 20 25 30 35 40 45
 0  6 12 18 24 30 36 42 48 54
 0  7 14 21 28 35 42 49 56 63
 0  8 16 24 32 40 48 56 64 72
 0  9 18 27 36 45 54 63 72 81
```

Tagasivaade

Slaidide alguses oli näide `(~X∈!X∘.×X)/X←2↓⍳N`

- Leiab kõik algarvud väiksemad kui `N`

Kuidas?

1. `⍳N` genereerib vektori arvudest `0,...,N-1`

Struktuurne programmeerimine

- Esialgne APL struktuurprogrammeerimise vahendeid ei omanud
- Hilisemad realisatsioonid on need juurde lisanud
  - if-then konstruktsioon
  - Tsüklid
  - Funktsioonide defineerimine
- Siin mõned näited A+ baasil

Struktuurne programmeerimine

Faktoriaali arvutamise rekursiivse funktsiooniga:

```
      fact{n}: if (n>0) n×fact{n-1} else 1
      fact{5}
 120
```

Faktoriaali arvutamine while-tsükliga:

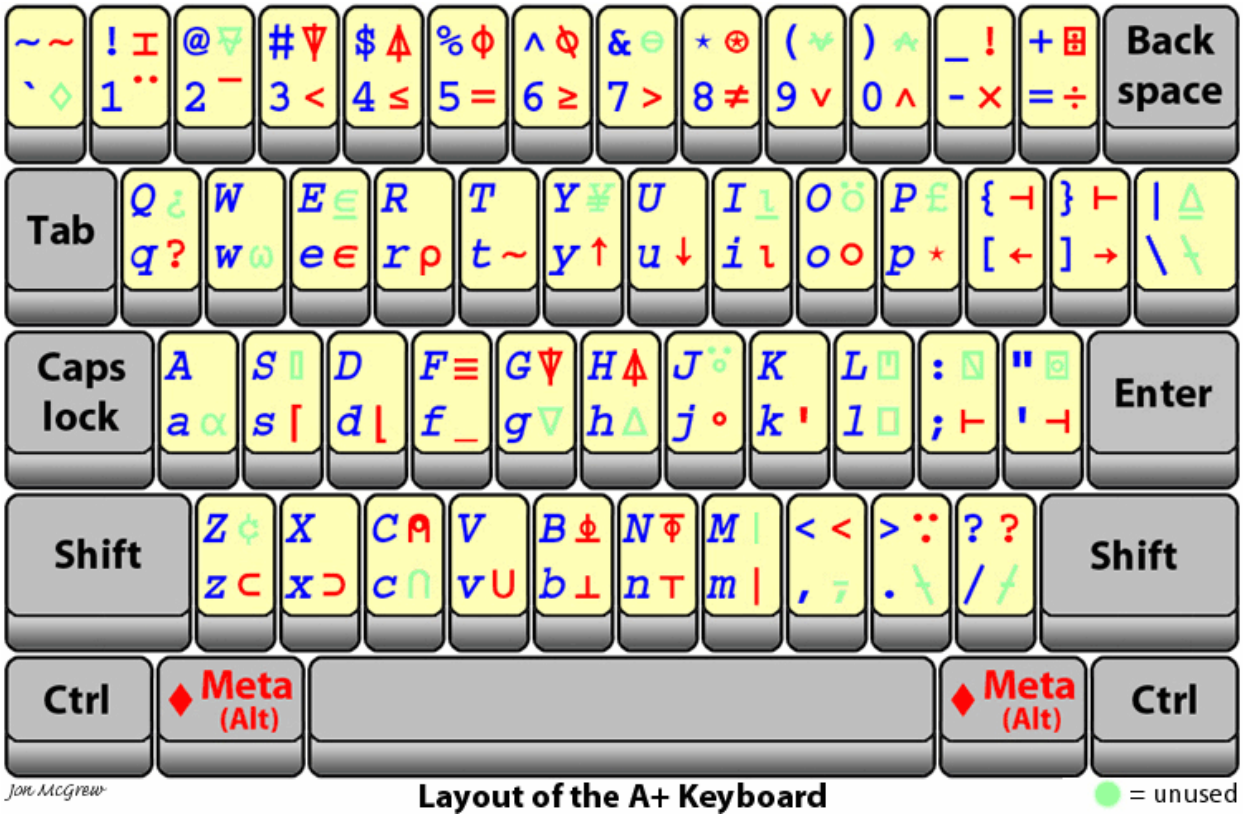
```
      f←1
      n←5
      while (n>0) { f←f×n; n←n-1 }
 0
      f
 120
```

A+ kasutamine

- Tarvis on aplus keskkonda ja sobivaid fonte
- Ubuntu
  - `apt-get install aplus-fsf-dev`
  - `apt-get install xfonts-kapl`
  - `xset fp rehash`
  - `xset fc cache`
- Terminali käivitamine: `xterm -fn *-kapl-*iso8859-1 -en 'ISO 8859-1'`
- A+ interpretaatori käivitamine: `a+`

A+ klaviatuur

- APL Union Keyboard
- Standardne APL paigutus



Kokkuvõte

APL:

- Andmemassiividel (array) programmeerimisele orienteeritud
- Üks vanemaid programmeerimiskeeli
- Realisatsioonid interpretaatori kujul
- Kasutab matemaatilist notatsiooni ja palju erimärke
- Arendusprotsess interaktiivses keskkonnas
- Suuremad APL programmid on loetamatud (*write-once-read-never*)
- Sobiv kasutamiseks statistika- ja finantsrakendustes