

Informatika pro moderní fyziky (6) výstupní a vstupní soubory pro výpočetní programy, datové struktury

František HAVLŮJ

e-mail: haf@ujv.cz

ÚJV Řež

oddělení Reaktorové fyziky a podpory palivového cyklu

akademický rok 2020/2021, 9. listopadu 2020

- 1 Co jsme se naučili minule
- 2 Automatizace tvorby vstupů
- 3 Automatizace tvorby vstupů – zobecnění
- 4 Načítání složitějšího výstupu

Obsah

- 1 Co jsme se naučili minule
- 2 Automatizace tvorby vstupů
- 3 Automatizace tvorby vstupů – zobecnění
- 4 Načítání složitějšího výstupu

- komplexní zpracování dat z výpočetních programů
- použití hashe jako univerzální datové struktury (místo pole)
- opakování: tvorba grafů s použitím gnuplotu

Obsah

- 1 Co jsme se naučili minule
- 2 Automatizace tvorby vstupů**
- 3 Automatizace tvorby vstupů – zobecnění
- 4 Načítání složitějšího výstupu

Velké množství podobných výpočtů

- vrátíme se na začátek problému – pracovali jsme s výstupními soubory, ale potřebujeme vyřešit provedení těchto výpočtů
- neboli vytvoření 11 x 11 vstupních souborů
- liší se pouze polohou tyčí R1 a R2
- pochopitelně nebudeme dělat ručně
- řešením je parametrizovaná šablona

Určení poloh tyčí

Ve vstupním souboru si najdeme relevantní část:

```
c -----  
c polohy tyci (z-plochy)  
c -----  
c  
67 pz 47.6000      $ dolni hranice absoberu r1  
68 pz 40.4980      $ dolni hranice hlavice r1  
69 pz 44.8000      $ dolni hranice absoberu r2  
70 pz 37.6980      $ dolni hranice hlavice r2
```

Výroba šablon

Jak dostat polohy tyčí do vstupního souboru? Vytvoříme šablonu, tzn nahradíme

```
67 pz 47.6000      $ dolni hranice absoberu r1
```


Výroba šablon

Jak dostat polohy tyčí do vstupního souboru? Vytvoříme šablonu, tzn nahradíme

```
67 pz 47.6000      $ dolni hranice absoberu r1
```

nějakou značkou (*placeholder*):

```
67 pz %r1%        $ dolni hranice absoberu r1
```

Vezměme to tak (i když to možná není pravda), že polohy jsou od 0 do 640 mm a že je to přímo kóta spodní hrany absorberu.

Chytáky a zádrhele

- kromě samotné plochy konce absorbéru je nutno správně umístit i z-plochu konce hlavice o 7,102 cm níže
- obecně je na místě ohlídat si, že placeholder nebude kolidovat s ničím jiným

Doporučené nástroje jsou:

- již známá funkce `sub / gsub` pro nahrazení jednoho řetězce jiným
- pro pragmatické lenochy funkce `File.read` načítající celý soubor do řetězce (na což nelze v mnoha programovacích jazycích ani pomyslet)
- možno ovšem použít i `File.readlines` (v čem je to lepší?)

Realizace

```
DELTA = 44.8000 - 37.6980

template = File.read("template")
(0..10).each do |i1|
  (0..10).each do |i2|
    r1 = i1 * 6.4
    r2 = i2 * 6.4
    s = template.gsub("%r1%", r1.to_s)
    s = s.gsub("%r1_%", (r1 - DELTA).to_s)
    s = s.gsub("%r2%", r2.to_s)
    s = s.gsub("%r2_%", (r2 - DELTA).to_s)
    File.write("inputs/c_#{i1}_#{i2}", s)
  end
end
```

Obsah

- 1 Co jsme se naučili minule
- 2 Automatizace tvorby vstupů
- 3 Automatizace tvorby vstupů – zobecnění**
- 4 Načítání složitějšího výstupu

A co takhle trochu zobecnění?

- když budu chtít přidat další tyče nebo jiné parametry, bude to děsně bobtnat
- funkce `process("template",
"inputs/c_{i1}_#{i2}", {'r1' => r1, 'r2'
=> r2,})`
- všechno víme, známe, umíme...
- rozšiřte tak, že třeba tyč B1 bude mezi R1 a R2, B2 mezi R1 a B1, B3 mezi R2 a dolní hranicí palivového článku ($Z = 1$ cm)

Obsah

- 1 Co jsme se naučili minule
- 2 Automatizace tvorby vstupů
- 3 Automatizace tvorby vstupů – zobecnění
- 4 Načítání složitějšího výstupu**

HELIOS

Tabulka výstupů:

```
List name      : list
List Title(s)  1) This is a table
                2) of some data
                3) in many columns
                4) and has a long title!
```

	bup	kinf	ab	ab	u235	
0001	0.00E+00	1.16949	9.7053E-03	7.6469E-02	1.8806E-04	7.
0002	0.00E+00	1.13213	9.7478E-03	7.9058E-02	1.8806E-04	7.
0003	1.00E+01	1.13149	9.7488E-03	7.9070E-02	1.8797E-04	7.
0004	5.00E+01	1.13004	9.7521E-03	7.9093E-02	1.8760E-04	7.
0005	1.00E+02	1.12826	9.7559E-03	7.9218E-02	1.8714E-04	7.
0006	1.50E+02	1.12664	9.7594E-03	7.9407E-02	1.8668E-04	7.
0007	2.50E+02	1.12399	9.7657E-03	7.9869E-02	1.8577E-04	7.
0008	5.00E+02	1.12007	9.7812E-03	8.1065E-02	1.8351E-04	7.
0009	1.00E+03	1.11561	9.8203E-03	8.3169E-02	1.7914E-04	7.
0010	2.00E+03	1.10542	9.9329E-03	8.6731E-02	1.7088E-04	7.
0011	3.00E+03	1.09354	1.0067E-02	8.9717E-02	1.6316E-04	7.
0012	4.00E+03	1.08126	1.0207E-02	9.2299E-02	1.5591E-04	7.
0013	6.00E+03	1.05755	1.0474E-02	9.6562E-02	1.4258E-04	7.

Co bychom chtěli

- mít načtené jednotlivé tabulky (zatím jen jednu, ale bude jich víc)
- asi po jednotlivých sloupcích, sloupec = pole (hodnot po řádcích)
- sloupce se nějak jmenují, tedy použijeme `Hash`
- `table["kinf"]`
- pozor na `ab`, asi budeme muset vyrobit něco jako `ab1`, `ab2` (ale to až za chvíli)

Nástrahy, chytáky a podobně

- tabulka skládající se z více bloků
- více tabulek
- tabulky mají jméno `list name` a popisek `list title(s)`

Jak uspořádat data?

- pole s tabulkami + pole s názvy + pole s titulky?

Jak uspořádat data?

- pole s tabulkami + pole s názvy + pole s titulky?
- co hashe tabulky[název] a titulky[název]?

Jak uspořádat data?

- pole s tabulkami + pole s názvy + pole s titulky?
- co hashe tabulky[název] a titulky[název]?
- nejchytřeji: `{"a"=>{:title => "Table title",
:data => {"kinf"=>...}}}`

Jak uspořádat data?

- pole s tabulkami + pole s názvy + pole s titulky?
- co hashe tabulky[název] a titulky[název]?
- nejchytřejši: `{"a"=>{:title => "Table title",
:data => {"kinf"=>...}}}`
- “nová” syntaxe: `{"a"=>{title: "Table title",
data: {"kinf"=>...}}}`

Z příkazové řádky

- a co takhle z toho udělat skript, který lze pustit s argumentem = univerzální
- `ruby read_helios.rb helios1.out`
- vypíše seznam všech tabulek, seznam jejich sloupců, počet řádků
- pole `ARGV` – seznam všech argumentů
- vylepšení – provede pro všechny zadané soubory: `ruby read_helios.rb helios1.out helios2.out` (tip: využijte vlastní metody, kde to jen jde)

Co jsme se naučili minule
Automatizace tvorby vstupů
Automatizace tvorby vstupů – zobecnění
Načítání složitějšího výstupu

A to je vše, přátelé!

