TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

# BAKALÁŘSKÁ PRÁCE

Liberec 2012

**Tomáš Košek**

**TECHNICKÁ UNIVERZITA V LIBERCI**
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

# Konvertor vstupních dat pro Flow123d

# Flow123d input data conversion tool

**Bakalářská práce**

Autor: **Tomáš Košek**
Vedoucí práce: Mgr. Jiří Vraný, Ph.D.
Konzultant: Mgr. Jan Březina, Ph.D.

**V Liberci 1. 5. 2012**

# Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

# Abstrakt


## Česká verze

Bakalářská práce je zaměřena na vytvoření konventoru vstupních dat pro program Flow123d. Jedná se o konfigurační ini soubor a soubor obsahující materiály. Oba soubory se uloží do stejného nového souboru ve formátu JSON. Na základě specifikací byl vytvořen konvertor v programovacím jazyce Python. Konventor se ovládá přes příkazovou řádku. Pomocí až osmi parametrů lze pozměnit průběh konverze. S konvertorem byly provedeny testy a výsledné soubory odpovídají požadavkům.



## Klíčová slova

Flow123d, Python, JSON, konvertor, ini.



## English version

This bachelor work is focused on creation of input data converter for program Flow123d. These are configuration ini file and file that contains materials. Both files are saved in a same new file in JSON format. Based on specifications converter was created in programming language Python. Converter is controlled via command line. Conversion process can be altered by up to eight parameters. Tests were made with this converter and result files corresponds to requirements.



## Keywords

Flow123d, Python, JSON, converter, ini.

# Obsah

## Seznam kapitol

# Seznam obrázků



# Seznam symbolů, zkratek a termínů

- JSON: JavaScript Object Notation. Jedná se o formát určený k výměně dat.
- Template soubor: vzorový soubor.
- Několik termínů z oblasti programovacích jazyků:
  - Int, integer: celočíselná hodnota.
  - Float, double: číslo s pohyblivou desetinnou čárkou.
  - String, řetězec: sled jednotlivých znaků včetně mezer.

- Bool, boolean: logická hodnota. Může být buď *true* (pravda) nebo *false* (nepravda).
- Null: absence jakékoliv hodnoty.

# 1 Úvod

Program Flow123d vznikl jako projekt několika geoinformatiků z ústavu nových technologií a aplikované informatiky (NTI) a jeho účelem bylo, aby dokázal nasimulovat proudění vody v podzemí. Mezi aktivní autory programu patří:

- Mgr. Jan Březina, Ph.D.
- Ing. Jiří Hnídek, Ph.D.
- Ing. Jiří Kopal
- Mgr. Jan Stebel, Ph.D.
- Ing. Jakub Šístek, Ph.D.
- Ing. Lukáš Zedek
- Ing. Vojtěch Wrnata

Mezi další autory programu patří:

- doc. Ing. Otto Severýn, Ph.D.
- doc. Ing. Dalibor Frydrych, Ph.D.
- doc. Ing. Milan Hokr, Ph.D.
- doc. Ing. Jan Šembera, Ph.D.

Hlavní výhodou programu je schopnost pracovat na velikém množství dat. Kromě soubory se samotnými daty využívá program konfigurační soubor, který obsahuje názvy ostatních souborů a také všechny potřebné parametry pro konverzi. Tento soubor byl až do verze 1.6.5 ve formátu ini.

Postupem času se ale začal hledat nový formát pro konfigurační soubor, který by byl lépe čitelný pro člověka a snadněji parsovatelný. S postupným vývojem programu přišla verze 1.7.0, která obsahovala nové konvence pro vstupní soubory. Došlo ke změně formátu z ini na formát JSON, který byl lehce upraven pro potřeby projektu (oproti původnímu formátu umožňoval například ukládat poznámky přímo). Na tento formát byl převeden i soubor, který obsahuje materiály (obvykle koncovka .mtr). Všechny změny a odlišnosti byly detailně popsány v projektové dokumentaci.

Po změně formátu ale zůstala celá řada souborů ve starém formátu a další soubory stále vznikaly. Z tohoto důvodu se rozhodlo, že bude potřeba vytvořit konvertor, který dokáže změnit stará data v ini formátu a data v materiálech na formát JSON. Současně bylo také potřeba, aby tento konvertor dokázal pozměnit strukturu podle požadavků tvůrců projektu.

Cílem této bakalářské práce tedy bylo vytvoření tohoto konvertoru. Jako programovací jazyk byl zvolen Python pro svoji velkou univerzálnost a přenositelnost mezi systémy. Protože je samotný program Flow123d spouštěn a ovládán z příkazové řádky, konvertor bude také pracovat z příkazové řádky. V budoucnu totiž může být potřeba spouštět vzniklý konvertor jako skript uvnitř samotného programu Flow123d. Konvertor bude muset také dokázat kromě správné konverze jednoduchých hodnot i celou řadu výjimečných případů.

# 2 Podrobnější představení problematiky

## 2.1 Flow123d

Flow123d je program, který slouží k simulaci vodních proudů a přesunu rozpuštěných látek v heterogenním porézním a rozpraskaném prostředí. Je určen především pro simulaci podzemních procesů v masách žulové skály. Program dokáže jednoznačně popsat procesy ve 3D prostředí, 2D frakturách a 1D kanálech a výměny mezi doménami různých dimenzí. Výpočetní síť je tedy kolekce 3D čtyřstěnů, 2D trojúhelníků a 1D úseček.

NOTE: možná trochu rozšířit

### 2.1.1 Vstupní soubory

Flow123d využívá až 8 vstupních souborů:

- Hlavní soubor (konfigurační): obsahuje názvy ostatních souborů a také všechny parametry nutné pro samotný výpočet pomocí programu Flow123d. Do verze 1.6.5 se používal formát ini. Od verze 1.7.0 se vybral jako nový formát JSON.
- 4 povinné soubory: obsahují data pro samotný výpočet. Mezi tyto soubory patří výpočtová síť (*.msh), soubor sousedností mezi dimenzemi (*.ngh), soubor s popisem materiálů (*.mtr) a soubor s hraničními podmínkami pro problém vodního toku (*.bcd).
- 3 nepovinné parametry: obsahují data pro některé části výpočtů. Mezi tyto soubory patří soubor s hraničními podmínkami pro transport (*.tbc), soubor s počátečními podmínkami pro jednotlivé instance transportu (*.tic) a soubor s vodními zdroji (*.fic).

### 2.1.2 Přechod na verzi 1.7.0

Důležitou součástí přechodu na novější verzi programu Flow123d bylo ustanovení nového formátu pro hlavní konfigurační soubor. V předchozích verzích se používal soubor ve formátu ini. Od této verze se začíná používat formát JSON. Ostatní vstupní soubory zůstávají nezměněny. Protože existuje mnoho konfiguračních souborů ve starém formátu a další nové vznikají, bylo rozhodnuto o vytvoření konvertoru, který dokáže zkonvertovat konfigurační soubory z formátu ini do formátu JSON.

Kromě změny formátu se změnily také adresy jednotlivých hodnot v konfiguračním souboru. Některé hodnoty je nutné přetypovat na jiný datový typ. Dále je nutné automaticky vytvořit několik nových hodnot, které se začínají používat od této verze.

## 2.2 JSON

**JSON** = **J**ava**S**cript **O**bject **N**otation. Jedná se o formát určený na výměnu dat mezi rozdílnými systémy a programovacími jazyky. Formát byl navržen tak, aby byl pro člověka snadno čitelný i zapisovatelný. JSON je také velmi snadno strojově analyzovatelný i generovatelný. JSON je založen na podmnožině programovacího jazyka JavaScript. Formát JSON je čistě textový a na jazyce nezávislý formát, využívá ale konvence z jazyků rodiny C, mezi které patří například C, C++, C#, Python, Perl, Java, Javascript a další.

Základem JSONu jsou dvě základní struktury:

- Kolekce párů název/hodnota. Jedná se o nesetřízený seznam hodnot. V programovacích jazycích bývá kolekce nazývána také názvem objekt, záznam, struktura, slovník, hash tabulka, klíčový seznam nebo asociativní pole.
- Tříděný seznam hodnot. Hodnoty zde nemají názvy, místo nich mají indexy. V programovacích jazycích toto odpovídá polím, vektorům, seznamům nebo posloupnostem.

### 2.2.1 Definice formátu

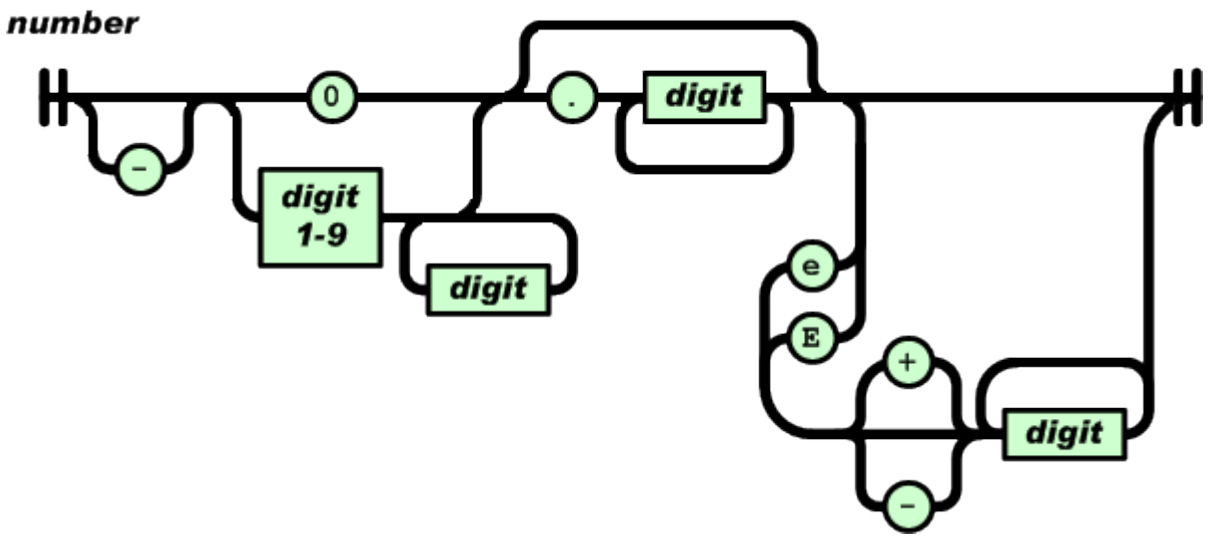
JSON se skládá z následujících elementů:

- Objekt (object): ohraničen složenými závorkami. Může obsahovat páry název/hodnota ve formátu „řetězec:hodnota“. Jednotlivé páry jsou od sebe odděleny čárkou.
- Tříděný seznam hodnot (Array): ohraničen hranatými závorkami. Může obsahovat hodnoty oddělené od sebe čárkou.
- Hodnota:
  - Řetězec
  - Číslo
  - Objekt
  - Tříděný seznam hodnot

- True / false
- null

- Řetězec: ohraničen uvozovkami. Může obsahovat únikové sekvence s použitím zpětného lomítka. Jediný znak je brán také jako řetězec.
- Číslo: může být ve formátu integer i double. Může obsahovat exponenciál. Nelze ale použít oktalový nebo hexadecimální zápis.

Všechny tyto definice pochází z domovské stránky pro JSON. Formát je velmi detailně popsán. Kromě obvyklé textové podoby je také popsán pomocí ilustrativních obrázků. Zde je například definice pro čísla:




*Obr. 1: Definice čísla ve formátu JSON*

## 2.3 Python

Python je silný a dynamický programovací jazyk, který je používán v širokém spektru aplikací. Mezi jeho největší výhody patří například jeho rychlost, množství standardních i stažitelných knihoven, zapustitelnost v aplikacích jako skriptovací rozhraní a přehlednost samotného kódu. Python má velice širokou komunitu a kompletní dokumentaci. Protože je Python pod open-source licencí, je možné distribuovat své výtvory v Pythonu i komerčně.

### 2.3.1 Python a JSON

Pro Python existuje celá skupina knihoven, které jazyku umožňují načítat a vytvářet soubory ve formátu JSON. Python obsahuje už v základním balení knihovnu *json*. Mezi stažitelné knihovny pro Python patří *simplejson* a *cjson*. Mezi těmito knihovnami je několik rozdílů:

- Knihovna *json*: je součástí standardních knihoven Pythonu. Obsahuje všechnu potřebnou funkcionalitu pro operace s JSONem. Pro použití je nutná minimální verze Pythonu 2.6. Nevýhoda této knihovny je fakt, že je aktualizována až s novou verzí Pythonu.
- Knihovna *simplejson*: je víceméně kopie standardní knihovny *json*. Oproti ní ale dokáže pracovat s Pythonem od verze 2.4. Jako externí knihovna je častěji aktualizována a bývá proto více optimalizovaná.
- Knihovna *cjson*: je mnohem rychlejší než předchozí dvě knihovny. Bohužel ale nedokáže občas správně dekódovat některé vstupy. Především když komunikuje s JSON modulem v PHP, přestává *cjson* fungovat. Opět jako externí knihovna je často aktualizována.

Protože konvertor nebude pracovat na příliš velkých datech (ztráta rychlosti tedy tolik nevadí) a bude nutné, aby bylo vše dekódováno zaručeně správně, vybral jsem standardní modul *json*.

### 2.3.2 Důležité metody knihovny json

Knihovna json obsahuje několik užitečných metod, ale v samotném konvertoru se používá z této knihovny pouze dvou metody. Jsou to metody *dumps* a *load*.

Metoda *dumps* vezme jako hlavní parametr objekt, který odpovídá strukturou poli nebo slovníku. Pokud je objekt validní, metoda vrátí čistý text ve formátu JSON. Metoda přijímá také celou řadu nepovinných parametrů. Je tedy možno výsledná data například seřadit abecedně nebo rozdělit na řádky pro větší přehlednost. V konvertoru se tato metoda využívá na závěr konverze, kdy se z výstupní proměnné získá potřebný text.

Metoda *load* slouží k nahrání dat ve formátu JSON do objektu (proměnné) v Pythonu. Lze tuto metodu také použít ke kontrole, zda je soubor validní. Metoda přijímá jako hlavní parametr soubor, který obsahuje data ve formátu JSON. V konvertoru se tato metoda používá po skončení konverze ini → JSON. Metodou se ověří validita dat. Získaná data jsou pak využívána pro konverzi.

# 3 Cíl práce

## 3.1 Přesné znění zadání práce

- Seznamte se s existujícími formáty vstupních souborů programu Flow123d, s návrhem nových formátů založených na JSON a dále s podporou formátu JSON v programovacím jazyce Python.
- Na základě získaných znalostí vytvořte návrh aplikace - konvertoru starých a nových formátů vstupu pro Flow123d.
- Návrh prakticky implementujte s využitím programovacího jazyka Python.

## 3.2 Upřesnění zadání práce

- Konvertor dat se bude vztahovat na dva z možných vstupních souborů pro Flow123d – konfigurační soubor ve formátu ini a oubor obsahující materiály (.mtr)
- Konvertor bude spouštěn a ovládán přes příkazovou řádku.
- Konvertoru bude možné přes parametry určit názvy vstupního a konvertovaného souboru.
- Výsledný konvertovaný soubor bude odpovídat zásadám formátu JSON.
- Konverzi bude možné pozměnit editováním template souboru, kde budou uloženy instrukce pro konverzi.
- Konvertor bude muset ošetřit všechny případné výjimky, které se budou ve vstupním souboru vyskytovat.
- Konvertor bude obsahovat vnitřní dokumentaci.
- Angličtina bude použita jako jediný jazyk (dokumentace i kód).
- Konvertor by měl obsahovat funkci, která pro větší přehlednost odmaže nepotřebné uvozovky u klíčů a přepíše dvojtečky na znak „=“.

Podrobný rozpis omezení a správných formátů je uveden na začátku další kapitoly. Stejně tak jsou popsány i jednotlivé výjimky, které se v konverzi mohou vyskytnout.

# 4 Návrh řešení

## 4.1 Požadovaná funkčnost programu

Cílem práce je vytvořit program ovladatelný z příkazové řádky, který dokáže ze souborů ve formátu ini převést informace do formátu JSON a upravit je podle přiloženého template souboru do požadovaného tvaru. Program bude kompletně v angličtině a musí obsahovat vnitřní dokumentaci. Konkrétní specifikace jednotlivých částí následují v dalších blocích.

### 4.1.1 JSON formát

JSON formát byl zvolen tvůrci programu Flow123d pro svoji snadnou přenositelnost mezi různými jazyky a jeho dobrou čitelnost pro člověka. Výsledný soubor proto musí být v korektním JSON formátu, aby ho dokázal parser programu Flow123d rozpoznat. Specifikace JSON formátu je uvedena v rešeršní části v podkapitole „JSON“.

### 4.1.2 Template soubor

Template soubor bude jednoduchý textový soubor, který bude obsahovat instrukce pro konverzi do JSON formátu. K tomuto souboru budou mít přístup pouze správci programu Flow123d. V případě změn v potřebách funkčnosti programu Flow123d musí být soubor jednoduše editovatelný bez nutnosti zásahu do kódu konvertoru. Struktura template souboru není pevně stanovená a bude definována tvůrcem konvertoru.

### 4.1.3 Standardní instrukce template souboru

Seznam základních možností pro template soubor:

- Změna adresy pro libovolnou hodnotu
- Vyhození (neuložení) hodnoty
- Změna datového typu hodnoty
  - Datové typy: string, int, double, bool, enum (výčtový datový typ)
- Přiřazení napevno stanovené hodnoty
- Tvorba pokročilé struktury v adrese (pole skupin hodnot)
- Psaní poznámek pro lepší orientaci správce programu Flow123d

### 4.1.4 Výjimečné instrukce template souboru

Seznam výjimek pro konverzi do finální podoby souboru (výjimka je označena jménem proměnné, u které je první výskyt výjimky):

- N_substances – tato hodnota nebude uložena do souboru, ale její hodnota bude určovat velikost pole v proměnné *Substances*
- New keys – všechny hodnoty v této sekci se uloží do souboru i v případě jejich absence ve zdrojovém souboru
- Transport_on – hodnota či absence této proměnné ovlivní zbytek hodnot v sekci *Transport*
    - *True* – hodnota transport_on se neuloží do souboru
    - *False* / absence hodnoty – adresa všech dalších hodnot v sekci *Transport* bude pozměněna přidáním „_save“ do posledního článku adresy (pro zachování dat)
- Decay – tato výjimka je příliš komplikovaná a proto bude popsána v dalším bloku

### 4.1.5 Speciální výjimka decay

V případě, že bude hodnota *Compute_decay* v sekci *Reaction_module* nastavena na true, provede se zpracování výjimky *decays*. Ze stejné sekce se také načtou hodnoty *Nr_of_decay_chains* a *Nr_of_FoR*, které udávají počet D*ecay* a *FoReact* oddílů. Jednotlivé oddíly mají k názvu připojený znak „_“ a jeho číslo (začínající od 1).

Všechna data se ukládají na jednotnou adresu do pole skupin hodnot. Pro každý oddíl *Decay_N* (*N* je pořadové číslo oddílu) se načtou jeho hodnoty a na jejich základě se provede uložení do výsledného souboru:

- Nr_of_isotopes = 2, Bifurcation_on = false
    - Ukládá se první hodnota *half_life* a pole *Substances* se 2 prvky
- Nr_of_isotopes > 2, Bifurcation_on = false
    - Uložení se rozloží na $N - 1$ jednotlivých záznamů.
    - Pro každý záznam i (pro všechna $0 <= i < N - 1$) se uloží *half_life*[*i*] a pole Substances s prvky *Substances*[*i*] a *Substances*[*i* + 1]
- Bifurcation_on = true

- Ukládá se první hodnota *half_life*, pole *Substances* a pole *Bifurcation*

Následně se prověří všechny oddíly *FoReact_N* (*N* je pořadové číslo oddílu). Každý oddíl se uloží do stejného pole jako *Decay* oddíly. Postup je podobný první možnosti pro *Decay* oddíly. Do výsledného souboru se ukládá *Kinetic_constant* a pole *Substances*.

### 4.1.6 Soubor s materiály

Data z tohoto souboru budou uložena na stejný soubor jako konvertovaný ini. Adresa souboru s materiály se vloží jako další parametr. Soubor obsahuje několik možných sekcí, ale konvertovat se bude pouze 6 z nich. Jednotlivé sekce začínají ve formátu *$Název* a jsou ukončeny *$EndNázev.* Ukázka je vidět na následujícím obrázku:

```
$Materials
  3
  48  11  10
  184 22  1 2
  256     33      1 2 3 0.1 0.2 0.3
$EndMaterials


$Storativity
  48  2.3
  184 3.4
$EndStorativity

$Geometry
        48      1       1.3
        184     2       1.4
$EndGeometry


$Sorption
  48  0 1 0.02
  184 0 1 0.02
  48  1 2 0.02  0.5
  184 1 2 0.02  0.5
$EndSorption
```

*Obr. 2: Ukázka souboru s materiály*

Jednotlivé sekce a jejich nové adresy, které se budou překládat:

- Materials – */problem/primary_equation/conductivity* ve formátu *material* a *value*. Podoba hodnoty *value* je určena druhým parametrem.

- Storativity – */problem/primary_equation/storativity* ve formátu *material* a *value*.
- Geometry – */problem/primary_equation/cross_area* ve formátu *material* a *value*. Hodnota value odpovídá třetímu parametru, druhý parametr se vynechává.
- Sorption – */problem/secondary_equation/sorption* ve formátu *material*, *substance*, *type* a *coef*. *Type* bude změněn na enumerický typ a jeho hodnota bude ovlivňovat počet hodnot v poli *coef*.
    - Type 1: Equilibrium
    - Type 2: Freundlich
    - Type 3: Langmuir
- Storativity – */problem/secondary_equation/sorption_fraction* ve formátu *material* a *value*.
- DualPorosity – */problem/secondary_equation/dual_porosity* ve formátu *material*, *mobile*, *immobile* a *coef*.

## 4.2 Základní návrh řešení

Programovací jazyk, který je na daný úkol velice vhodný, je Python. Python má v sobě implementovanou knihovnu *json*, která velice usnadní případnou práci se soubory ve formátu JSON. Dalším důvodem je multiplatformní povaha Pythonu.

Protože knihovna json může pracovat jen se soubory ve formátu JSON a nebo s některými objekty, byla zapotřebí před upravováním adres hodnot a ošetřování výjimek nejdříve získat data v JSON formátu. Proto se konverze rozdělila do 3 fází. První fáze řádek od řádku převádí ini soubor do podoby, která odpovídá standardům JSON formátu. Druhá fáze poté tyto snadno získatelné data využívá pro úpravy do finální podoby souboru. Před uložením finálního souboru se ještě může spustit třetí fáze: konverze souboru s materiály. Pro lepší přehlednost se kromě finálního souboru ukládá také soubor obsahující čistou konverzi do JSON formátu.

## 4.3 Návrh funkčnosti modulů

Moduly byly rozděleny podle jejich funkčnosti. Ke každému modulu se také vytvořil odpovídající testovací modul pro potvrzení funkčnosti metod a rychlejšího odhalování chyb

v jednotlivých metodách. Program se tedy skládá ze čtyř modulů (plus testovací moduly). Jednotlivé úlohy modulů jsou popsány v dalších podkapitolách.

## 4.4 Hlavní modul

Tento modul má na starosti obsluhu celého programu. Jako kontrolní modul přijímá vstupy od uživatele a na jejich základě spouští jednotlivé fáze konverze. Po uživateli je žádáno pro spuštění několik parametrů: adresa ini souboru pro konverzi, název čisté konverze do JSON formátu a název finálního souboru. Volitelně lze přidat parametr, jestli se mají ukládat komentáře (standardně zapnutá funkce). Další volitelný parametr je zobrazování varovných hlášení při druhé fázi (standardně se nezobrazují). Program musí dokázat ošetřit absenci některých parametrů pozměněnou funkcionalitou nebo chybovým varováním na výstup konzole.

## 4.5 Modul pro první fázi

Účelem tohoto modulu je provádět čistou konverzi z ini formátu do JSON formátu. Program bude postupně procházet jednotlivé řádky ini souboru a bude je převádět do JSON formátu. Program si musí udržovat v paměti několik informací a na jejich základě upravuje vznikající JSON soubor:

- Úroveň zanoření konkrétního údaje – ovlivňuje počet tabelátorů (lepší čitelnost)
- Jestli se jedná o první sekci – nutnost ukončit nejdříve předchozí sekci
- Jestli se jedná o první hodnotu v sekci – vložení čárky na konec předchozí hodnoty
- Jaký byl poslední element – způsob ukončení sekce či hodnoty
- Komentář – přidání komentáře jako novou hodnotu (JSON nepodporuje komentáře přímo, proto se musí přidat jako nová hodnota)

Modul má ošetřené možné výjimky v podobě špatně zadaného názvu vstupního souboru. V takové případě se uživateli objeví varovné hlášení. Výstupní soubor musí být také ve správném formátu. V případě chyby se opět objeví uživateli varovné hlášení.

Program pro každou čtenou hodnotu zjišťuje její datový typ. V případě číselných datových typů se kontroluje validita jejich zápisu. V případě řetězců se doplňují uvozovky, aby hodnotu dokázal JSON parser rozeznat. Zdrojové soubory obsahují několik možných zápisů boolean hodnot, proto musí program tyto různé zápisy sjednotit na standardní *true*/*false*. Soubory také obsahují pole,

které jsou ve výsledném souboru ohraničené hranatými závorkami a jednotlivé prvky odděleny čárkou.

## 4.6 Modul pro druhou fázi

Tento modul provádí úpravy JSON souboru do finální podoby. Program postupně prochází jednotlivé řádky template souboru, který obsahuje jednotlivé instrukce pro konverzi. Na základě instrukcí modul postupně ukládá hodnoty na určené adresy a zbavuje se nepotřebných hodnot. V případě, že program nenalezne danou hodnotu ve zdrojovém souboru, je hodnota ignorována a uživateli se zobrazí upozornění. Template soubor stanovuje i datový typ hodnot, kde je kromě jednoduchých datových typů i složitější, jako například enum. Pomocná funkce tedy dokáže rozložit řetězec s možnostmi a porovnat je se skutečnou hodnotou. Modul také kontroluje výjimky spojené s touto konverzí:

- N_substances – modul si ukládá tuto hodnotu a pomocí ní omezuje velikost pole v hodnotě *Substances*
- New keys – modul si ukládá tyto hodnoty, i když je nenalezne ve zdrojovém souboru.
- Transport_on – Hodnota je uložena pouze pokud je *false*. Program si ale tuto hodnotu pamatuje v každém případě. Pro *false* se také upravují adresy všech ostatních hodnot v sekci *Transport*.

Poslední výjimka decays se příliš odlišuje od zbytku souboru. Proto byla pro tuto výjimku vytvořena speciální metoda. Ta se spustí po ukončení celé konverze. Následně lze ještě spustit konverzi souboru s materiály.

Přesný formát template souboru se postupně upřesňoval s vývojem programu a požadavky správců programu Flow123d.

## 4.7 Modul pro třetí fázi

Tento modul se stará o zpracování souboru s materiály a jeho začlenění do hlavního souboru. Nejdříve program musí nalézt umístění souboru. Toho dosáhne buď získáním hodnoty z ini souboru nebo přímým vstupem od uživatele. Program poté postupně prochází jednotlivé řádky souboru s materiály. Na základě toho, v jaké sekci se nachází, ukládá program data do polí. Výsledná pole jsou uložena do hlavního datového souboru, který je pak poslán zpět pro finální zpracování.

# 5 Realizace řešení

## 5.1 Popis programu

Konvertor má velmi jednoduchou obsluhu. Program pro svůj chod potřebuje některé ze čtyř hlavních a čtyř vedlejších parametrů popsaných v podkapitolách „Modul Converter.py“ a „Používání konvertoru“. Po přijetí parametrů program nejdříve zkonvertuje po řádcích ini soubor do formátu JSON. Tento vzniklý soubor se uloží a je možné s ním pracovat. Pro použití v programu Flow123d je ale nutné pozměnit částečně strukturu, proto se po této konverzi spustí druhá. Ve druhé fázi se pak upraví tento soubor podle instrukcí uložených na template souboru, ke kterému má přístup správa programu Flow123d. Po skončení konverze se ještě může spustit konverze souboru s materiály.

Popis struktury template souboru lze nalézt v podkapitole „Template soubor“. Tento konvertor byl vytvořen pro příkazovou řádku, ze které se ovládá a na které se objevuje výpis z průběhu konverze.

## 5.2 Struktura programu

Program je naprogramovaný v jazyce Python 2.7. Samotný program je složen ze čtyř normálních modulů a dvou testovacích modulů. Moduly programu jsou: *converter.py*, *conversion.py*, *completion.py* a *materials.py*. Testovací moduly byly vytvořeny k modulům *conversion.py* a *completion.py*. Zde následuje stručný přehled účelu modulů, detaily modulů jsou v příslušných podkapitolách.

Modul *converter.py* je hlavní modul, který ovládá chod programu a kontroluje vstupní data. Na základě vstupních dat pak spouští buď přímou konverzi ini souboru nebo úpravy souboru v JSON formátu podle template souboru.

Modul *conversion.py* se stará o konvertování ini souboru do JSON formátu. Postupně prohlíží řádky ini souboru a současně vytváří nový soubor v JSON formátu. Může selhat v případě špatně zadaných jmen souborů.

Modul *completion.py* provádí finální úpravy. Podle jednotlivých instrukcí uložených na template souboru vytváří pozměněnou kopii JSON souboru. Může selhat v případě špatně zadaných jmen souborů.

Modul materials.*py* provádí konverzi souboru s materiály. Na základě jednotlivých sekcí v souboru se doplňují data do hlavního souboru. Může selhat v případě špatně zadaného jména souboru.

## 5.3 Vstupní parametry

Konvertor je program pracující z příkazové řádky. Uživatel zadá jako parametry název ini souboru určeného ke konvertování, název přímé konverze ini souboru ve formátu JSON, název finálního souboru opět ve formátu JSON a název souboru s materiály. Parametry nejsou vždy povinné a jejich částečná absence programu nevadí. Mezi další možné parametry patří možnost vypnout ukládání komentářů, možnost zobrazovat hlášení o chybějících hodnotách, zapnutí debug módu a vypnutí finálních úprav souboru.

Pokud chybí první parametr, vynechává se přímá konverze do formátu JSON a spustí se úpravy podle template souboru. Pokud chybí druhý nebo třetí parametr (ale první je uveden), název souboru je stejný až na koncovku .json. Soubor s přímou konverzí má navíc název pozměněn o „_temp“. V případě absence prvních dvou parametrů se konverze nespustí, pouze upozorní uživatele na problém. Pokud chybí čtvrtý parametr, nespustí se konverze souboru s materiály.

## 5.4 Modul converter.py

Hlavní modul ovládající chod programu. Kontroluje vstupy z příkazové řádky a spouští jednotlivé konverze. Celkově přijímá až 8 vstupních parametrů:

- -ini – název nebo adresa zdrojového souboru v ini formátu. V případě, že už uživatel má zdrojový soubor ve formátu JSON, není nutné tento parametr vyplnit. Pokud je parametr uveden, provádí se první fáze konverze (ini → JSON).
- -json – název nebo adresa zdrojového souboru ve formátu JSON. Tento parametr je povinný pouze pokud chybí první parametr. V případě absence obou prvních parametrů se program zastaví. Pokud parametr chybí a první je přítomen, vytvoří se název tohoto souboru automaticky z názvu ini souboru.
- -final – název nebo adresa souboru ve finální podobě. V případě absence parametru se doplní automaticky z jednoho ze specifikovaných souborů (přednostně z ini souboru).
- -mat – název nebo adresa souboru s materiály. Pokud je tento parametr nastaven na „*“, pokusí se program získat adresu z ini souboru.

- -nc – při vyplnění tohoto parametru se zruší funkce ukládání komentářů.
- -sw – při vyplnění se zobrazují varovná hlášení o nenalezených hodnotách ze seznamu na template souboru.
- -db – zapnutí debug módu. Každá uložená či zahozená hodnota se vypíše na konzoli.
- -nd – vypnutí úprav, které se provádějí po skončení konverze (odstraňování uvozovek a nahrazení dvojtečky znakem =).

Po přijmutí parametrů se provedou případná doplnění jmen a pokud nenastala chyba (chybí první dva parametry), spustí se první nebo druhá fáze konverze (záleží na parametrech).

## 5.5 Modul conversion.py

Tento modul provádí první fázi celkové konverze: převedení ini souboru do formátu JSON. Program pracuje po jednotlivých řádcích, které pozměněné ukládá do výstupního souboru. Pokud neexistuje vstupní ini soubor nebo byl zadán neplatný název výstupního souboru, konverze se ukončí. Metody a jejich funkce jsou popsány v následujících blocích. Na obrázku je vidět proměna ini souboru do JSON formátu.

```
[Transport]
Transport_on        = Yes
Sorption        = Yes
Dual_porosity       = Yes
Concentration       = ./input/test2.tic
Transport_BCD       = ./input/test2.tbc
Transport_out         = ./transport.msh
Transport_out_im        = NULL
Transport_out_sorp        = NULL
Transport_out_im_sorp     = NULL
N_substances          = 2
Substances        = A B C D E
Substances_density_scales = 1 1 1
```

```
"Transport" : {
  "Transport_on" : true,
  "Sorption" : true,
  "Dual_porosity" : true,
  "Concentration" : "./input/test2.tic",
  "Transport_BCD" : "./input/test2.tbc",
  "Transport_out" : "./transport.msh",
  "Transport_out_im" : null,
  "Transport_out_sorp" : null,
  "Transport_out_im_sorp" : null,
  "N_substances" : 2,
  "Substances" : ["A", "B", "C", "D", "E"]
  "Substances_density_scales" : [1, 1, 1]
},
```

*Obr. 3: Část ini souboru zkonvertovaná do JSON formátu*

### 5.5.1 convert (adr, res, comments)

Hlavní metoda konverze. Jako vstupní parametry se přijímá adresa nebo název ini souboru (string), název nového souboru v JSON formátu (string) a nakonec jestli se mají zachovat komentáře (boolean, v současné době nepoužívaný prvek). Nejdříve se zkontroluje existence vstupního souboru. Pokud je soubor nalezen, zkontroluje se správnost názvu výstupního souboru.

Pokud je název správně, otevře se vstupní ini soubor a začne konverze. Metoda vrací *false* pokud nastala chyba v názvech souborů a *true* pokud konverze proběhla v pořádku.

Jednotlivé konvertované řádky ukládá metoda do výstupního souboru. Metoda postupně prochází jednotlivé řádky ini souboru a vyhodnocuje, co je na daném řádku typově za obsah. Možnosti jsou tři: oddíl, hodnota a nebo prázdný řádek či komentář. V případě prázdných řádků a komentářů se neprovádí nic a program pokračuje dalším řádkem.

Pokud se jedná o oddíl, zavolá se metoda *read_header*, která vrací nazpět jméno oddílu a ukončení předcházejících oddílů. Po začátku oddílu se zvýší počet tabulátorů pro další zápis, aby bylo dosaženo lepší čitelnosti. Jednotlivé oddíly jsou pak od sebe odděleny čárkou.

V případě hodnoty se nejdříve zjistí, jestli se jedná o první hodnotu v oddílu. Pokud není, tak se ukončí předchozí hodnota čárkou a pokračuje se na novém řádku. Pro správný zápis se zavolá metoda *read_data*, která navrátí hodnotu ve správném formátu. Pokud není zrušená funkce ukládání komentářů, tak se uloží komentář jako další hodnota s podobným názvem, jaký měla hodnota před komentářem (k názvu přidá „_comment“) a komentář se stane hodnotou této nové proměnné.

### 5.5.2 read_header (text)

Tato metoda slouží k získání jména oddílu. Přijímá jediný parametr: řádek programu bez prvního znaku označujícího oddíl (string). Metoda se současně stará o ukončení předcházejících oddílů, pokud nějaké byly. Návratová hodnota obsahuje možné ukončení předchozího oddílu a začátek nového oddílu (string).

### 5.5.3 read_data (text)

Tato metoda sestaví řádek s hodnotou do požadovaného tvaru. Přijímá jediný parametr a to řádek z ini souboru (string). Navrací pak řádek ve správném formátu (string). Metoda nejdříve rozdělí řádek na dvě části: název hodnoty a samotná hodnota. Od hodnoty se také oddělí případný komentář. Hodnota se pak může pozměnit zavoláním metody *change_words*, která je popsána níže. Zjistí se datový typ hodnoty skrze metodu *which_type* a podle výsledku se provede jeden ze 4 možných zápisů celé hodnoty:

- String – k hodnotě se přidají navíc uvozovky
- Int nebo Float – odstraní se nadbytečné nuly (JSON s nimi nedokáže pracovat)
- Boolean nebo null – hodnota se zapíše bez úprav
- Array – zavolá se funkce *read_array*, která vrátí správný výsledek

### 5.5.4 change_words (word)

Slouží ke změně konkrétních klíčových slov na jiná. Metoda přijímá jako jediný parametr hodnotu (datový typ je neznámý). Metoda navrací zpět tuto hodnotu v nezměněném datovém typu. Pokud hodnota odpovídá jednomu z klíčových slov, změní se na jiné předem určené slovo. V současné době se používá na změnu *on*/*off* a *yes*/*no* na *true*/*false*.

### 5.5.5 which_type (value)

Slouží k rozeznání datového typu hodnoty. Metoda jako vstupní parametr přebírá samotnou hodnotu (datový typ je neznámý). Návratová hodnota označuje datový typ této hodnoty (int). Metoda nejdříve zkontroluje, zda je hodnota číslo. Pokud není, zkontroluje se, jestli se hodnota nerovná *true*/*false*. Také se porovná s heslem *null*. Pokud ani to neprojde, zjistí se počet mezer v řetězci. Pokud zde nejsou mezery, jedná se o string. Pokud tam mezery jsou, jedná se o array.

### 5.5.6 read_array (text)

Účel této metody je zpracování polí. Vstupní parametr je pole hodnot oddělených mezerou (string). Výstupní parametr je pole zapsané ve správném tvaru (string). Metoda rozdělí řetězec podle mezer na jednotlivé hodnoty. Potom projde postupně tyto hodnoty a pro každou skrze metodu *which_type* určí její typ a podobně jako v metodě *read_data* provede zápis. Jednotlivé hodnoty jsou od sebe odděleny čárkou.

### 5.5.7 tabs ()

Tato metoda slouží k vypsání správného počtu tabulátorů v závislosti na úrovni v dokumentu. Nepřebírá žádný parametr a navrací správný počet tabulátorů (string). Tato metoda je volána z většiny ostatních metod na začátcích řádků.

## 5.6 Modul completion.py

Tento modul provádí druhou fázi celkové konverze: upravení vzniklého JSON souboru do potřebného tvaru podle instrukcí uložených v template souboru. Pokud neexistuje vstupní JSON soubor nebo byl zadán neplatný název výstupního souboru, konverze se ukončí. Metody a jejich funkce jsou popsány v následujících blocích. Tato metoda také používá knihovnu *json*.

### 5.6.1 complete (adr, res, warn, comm, dbg, alt, mat)

Hlavní modul starající se o spouštění jednotlivých částí úprav. Metoda má sedm vstupních parametrů: adresu vstupního JSON souboru převedeného z ini souboru (string), název finálního

upraveného JSON souboru (string), zobrazování varování pro chybějící hodnoty (boolean), ukládání komentářů (boolean), debug mód (boolean), upravování po skončení (boolean) a adresu souboru s materiály (string). Podobně jako hlavní metoda z *conversion.py* vrací tato metoda *true* pro povedenou konverzi a *false* pro selhání. Selhání může nastat v případě, když neexistuje vstupní soubor nebo je název výstupního souboru ve špatném formátu.

Metoda si skrze knihovnu json načte vstupní soubor do pomocné proměnné, která díky tomu obsahuje všechna důležitá data z JSON souboru pomocí knihovny json (metoda *json.load*). Současně se vytvoří nový soubor pojmenovaný podle vstupního parametru. Pokud už takový soubor existuje, tak se kompletně přepíše. Nakonec se otevře template soubor (soubor je popsán v další podkapitole), který obsahuje potřebné instrukce pro vytvoření finálního souboru. Načtená data se zpracují funkcí *read_template,* která navrátí data zpracovaná do výstupní proměnné.

Po skončení provádění instrukcí template souboru se zkontroluje, jestli chce uživatel používat hodnoty oddílu *Decay.* Pokud je hodnota *true*, pozmění se výstupní soubor pomocí speciální metody *decays*, protože tento oddíl není prakticky v ničem standardizovatelný se zbytkem template souboru.

Pokud je zadána adresa s materiály, spustí se metoda *start* modulu *materials.py.* Ta se postará o konverzi souboru s materiály a navrátí doplněnou výstupní proměnnou.

Nakonec se z výstupního souboru přes knihovnu JSON (metoda *json.dumps*) vygeneruje kód v JSON formátu. Tato textová data se ještě pozmění skrze funkci *array_check*, kde se opravuje vzniklá chyba s poli skupin proměnných. Pokud nejsou zakázány finální úpravy, tak se zde také provedou. Po skončení se upravená textová data nahrají do otevřeného výstupního souboru a uloží se.

### 5.6.2 read_template (template, d_in, warn, comm, dbg)

Tento modul se stará o zpracování instrukcí v template souboru. Metoda má pět vstupních parametrů: obsah template souboru v textové podobě (array), data získaná z JSON souboru (array), zobrazování varování pro chybějící hodnoty (boolean), ukládání komentářů (boolean) a debug mód (boolean).

Soubor se prochází postupně řádek po řádku a kontroluje se jeho obsah. Podobně jako ini soubor je template rozdělen do oddílů, takže si algoritmus pamatuje poslední navštívený oddíl. Kromě oddílu může být na řádku už jen instrukce nebo komentář. U instrukce se nejdříve zjistí, které hodnoty se týká. To se provede rozdělením řádku podle znaku „=“. První část vzniklého pole

obsahuje název hodnoty. Pokud je délka pole 2, jedná se zpravidla o to, že daná proměnná je zbytečná a do výstupu se neukládá. Současně se kontroluje první výjimka u hodnoty *N_substances* (vyjímky jsou popsány v předešlé kapitole). Tato proměnná se neuloží do výsledného souboru, ale zapamatuje se její hodnota.

Pokud je délka pole větší než 2, jedná se o pokročilejší instrukci. Nejdřív se program pokusí najít odpovídající proměnnou ve vstupním souboru. Pokud není nalezena a je zapnuta funkce zobrazování varování, je absence hodnoty oznámena uživateli na výstupu. Bez nalezené proměnné pokračuje program další instrukcí s jedinou výjimkou. Pokud se instrukce nalézá v oddílu „New keys“, je absence proměnné ignorována a program pokračuje dál, jako kdyby hodnota existovala.

Po nalezení hodnoty proměnné se kontroluje výjimka *Transport_on*: pokud je na řadě tato instrukce, tak se kontroluje hodnota této proměnné ve vstupním souboru a na základě toho se pracuje s ostatními instrukcemi v oddílu *Transport*. V případě hodnoty *true* se proměnná neukládá a zbytek oddílu *Transport* se provede normálně. Pro *false* se proměnná *Transport_on* uloží a zbytek oddílu je uložen se změněnou adresou, aby uživatel nepřišel o tato data.

Další kontrolovaná výjimka je pro proměnnou s názvem *Substances*. Protože hodnotou *Substances* je pole, tak se omezí velikost pole na základě uložené hodnoty z *N_substances*. Pokud tato hodnota nebyla nalezena, délka pole je 0 a díky tomu se ani neuloží.

Po těchto výjimkách se z instrukce vyparsuje nová adresa pro proměnnou. Adresa se nalézá v druhé části řádku (rozděleného skrze „=“) a začíná od uvozovky. V případě oddílu *Transport* může být tato adresa upravena.

Následně se ze třetí části instrukce vyseparuje datový typ. Hodnota je pak skrze metodu *value_type* pozměněna. Jako parametr této metody poslouží právě získaný datový typ. Se získanou hodnotou ve finální podobě se upraví výstupní proměnná pomocí funkce *update_data*. Pokud je aktivován debug mód, tak se uložení hodnoty vypíše na konzoly.

### 5.6.3 value_type (val, val_type)

Metoda v závislosti na typu proměnné upraví samotnou hodnotu proměnné. Přijímá dva parametry: hodnotu proměnné (neznámý datový typ) a typ proměnné (string). Výstup je proměnná v definovaném datovém typu (neznámý datový typ).

Metoda nejdříve zkontroluje, jestli se datový typ nerovná některému z jednoduchých datových typů jako například double, string, int nebo bool. V tom případě funkce navrátí hodnotu v nalezeném datovém typu. Pokud datový typ neodpovídá žádnému z těchto čtyř, prohledá se

začátek řetězce datového typu. V případě, že řetězec začíná heslem data, nastaví se hodnota proměnné na hodnotu nalézající se v závorkách za tímto slovem.

V případě, že řetězec začíná heslem enum, uloží se obsah uvnitř závorek do pomocné proměnné a skrze rozdělování přes znak čárky se vytvoří pole možností. Postupně se toto pole projde a každý segment se rozdělí podle znaku „>“ na dvě poloviny. Pokud se hodnota vstupní proměnné rovná levé polovině segmentu, tak je nahrazena druhou polovinou segmentu a cyklus se ukončí. V případě, že metoda nenašla shodu v enum nebo datový typ neodpovídal žádnému z předešlých možností, je navrácena původní hodnota.

### 5.6.4 update_data (value, sections, d_out)

Tato metoda slouží k ukládání nových dat do výstupní proměnné. Jako vstup přebírá hodnotu (neznámý datový typ), novou adresu proměnné (array) a výstupní soubor (array). Výstup metody je upravený výstupní soubor (array).

Metoda postupně prochází adresu a kontroluje, zda ve výstupním souboru jednotlivé části adresy existují. Při neexistenci jsou vytvořeny. Posledním segmentem adresy je samotná proměnná. Po uložení proměnné metoda navrátí takto upravenou proměnnou. V současném provedení je adresa limitována na 5 částí.

### 5.6.5 array_check (data, alt)

Metoda slouží k opravení chyby, která vzniká při provádění instrukcí a která by byla příliš komplikovaná na odstranění už během úprav. Metoda přijímá jako vstupní parametry celý výstupní soubor v textové podobě (string) a příkaz k provádění finálních úprav (boolean). Návratová hodnota je opět tento soubor opravený o vzniklé chyby v polích (string) a případně upravený.

Při úpravách je možné do nové adresy zapsat místo jednoho segmentu číslo a tím naznačit, že se jedná o pole skupin hodnot. Po překladu se číslo zapíše do adresy jednoduše jako řetězec (například „1“). Tato chyba se odstraní právě až po skončení úprav na samotném textovém souboru.

Postupně se prochází soubor po řádcích a hledá se článek adresy sestávající pouze z číslice. Jakmile se takový řádek najde, tak se odmaže číslice a nechá se jen složená závorka. Zároveň se uloží úroveň hloubky pole (jak hluboko je v adrese) a pomocná proměnná se nastaví na *true*. Na řádku před tímto prvním výskytem se změní složená závorka na hranatou. Poté se prochází zbytek řádků a hledají se další číselné klíče a provádí se u nich ale už jen odmazání číslice. Jakmile se úroveň přiblíží víc ke kořenům pole, přepíše se opět složená závorka na hranatou a pomocná proměnná se vrátí zpátky na *false*. Na obrázku je ukázán rozdíl v datech před a po úpravě.

```
"system": {
  "output_streams": {
    "1": {
      "file": "./test2.msh",
      "format": "gmsh_ascii",
      "name": "flow_output_stream"
    },
    "2": {
      "file": "./transport.msh",
      "name": "transport_output_stream"
    }
  },
  "pause_after_run": false
},
```

```
"system": {
  "output_streams": [
    {
      "file": "./test2.msh",
      "format": "gmsh_ascii",
      "name": "flow_output_stream"
    },
    {
      "file": "./transport.msh",
      "name": "transport_output_stream"
    }
  ],
  "pause_after_run": false
},
```

*Obr. 4: Zápis před a po opravení metodou array_check*

Pokud je povoleno upravování, tak se na každém řádku také změní „:“ na „=“. U klíčů, které neobsahují mezery, se odmažou uvozovky.

### 5.6.6 decays (res, data)

Tato metoda slouží k uložení informací oddílu *Decay*. Jako vstupní parametry přijímá proměnnou obsahující JSON data (array) a výstupní proměnnou (array). Po uložení nových informací metoda navrací upravenou výstupní proměnnou (array). Tento oddíl se zapisuje velice odlišně od zbytku dat a proto bylo potřeba vytvořit pro ni speciální metodu.

Program umožňuje si skrze template soubor zvolit adresu modulu a názvy jednotlivých proměnných v tomto oddíle. Proto se nejdříve načte template soubor. Postupně se prohledávají řádky, dokud se program nedostane do sekce *Settings*, kde jsou uložené jednotlivé názvy a adresa pro modul *Decays*. Data se poté ukládají na nastavitelnou adresu do pole skupin hodnot pod nastavitelné názvy.

Metoda nejdříve načte jednotlivé počty oddílů *Decay* a *FoReact*. Poté se postupně v cyklu přistupuje k jednotlivým sekcím. Program začíná u oddílů *Decay*. Nejdříve se načte hodnota/y proměnné *Half_lives*, *Nr_of_isotopes* (následně značena jako $N$) a pole *Substance_ids*. Poté se načte hodnota *Bifurcation_on*. Program se následně dělí na základě této hodnoty:

- Pokud hodnota chybí a nebo je nastavena na hodnotu *false*, tak se spustí cyklus o $N-1$ krocích. Pro každý krok i (počítáno od 0) se uloží do výstupního souboru do proměnné *Half_life* hodnota *Half_lives*[$i$] a do pole *Substances* se uloží dvojice hodnot *Substances_ids*[$i$] a *Substances_ids*[$i$+1].

- Pokud je hodnota nastavena na *true*, tak se do výstupního souboru uloží proměnná *Half_life* odpovídající první hodnotě proměnné *Half_lives*, dále se uloží pole *Substances* odpovídající poli *Substances_ids* a nakonec se uloží pole *Bifurcation* odpovídající poli stejného názvu.

Po posledním oddílu *Decay* program pokračuje oddíly *FoReact*. Tyto oddíly jsou jednodušší. Pro každý záznam se uloží do výstupní proměnné hodnota *Kinetic_constant* odpovídající hodnotě stejného názvu a pole *Substances* odpovídající poli *Substance_ids*. Výsledný soubor se navrátí.

## 5.7 Modul materials.py

Tento modul provádí třetí fázi celkové konverze: zkonvertování souboru s materiály do hlavního json souboru. Pokud neexistuje soubor s materiály, konverze se ukončí. Metody a jejich funkce jsou popsány v následujících blocích. Tato metoda také používá metodu *value_type* z modulu *completion.py.*

### 5.7.1 start (data, adress, input_adress)

Hlavní modul starající se o spouštění jednotlivých částí úprav. Metoda má tři vstupní parametry: výstupní proměnná (array), adresa souboru s materiály (string) a absolutní adresa konvertovaného souboru (string). Metoda vrací nazpět výstupní proměnnou obohacenou o data ze souboru s materiály. Selhání může nastat v případě, když neexistuje soubor s materiály.

Metoda postupně prochází jednotlivé řádky souboru a na základě jejich typu se provádí příslušná akce:

- Řádek začínající znakem „#“ je považován za komentář a je ignorován. Stejně tak řádky obsahující pouze netisknutelné znaky.
- Řádek ve formátu *$Název* nastaví oddíl na konkrétní název.
- Řádek ve formátu *$EndNázev* nastaví oddíl na prázdný řetězec. Jakékoliv datové hodnoty jsou ignorovány, dokud nezačne nový oddíl.
- Datové hodnoty jsou na základě současného oddílu zpracovány do příslušného pole hodnot. Pro některé oddíly se volá speciální metoda, které se jako parametr posílá pole hodnot konkrétního řádku.
    - Řádky v oddílu *Materials* jsou zpracovány skrze metodu *sec_materials*.
    - Řádky v oddílu *Sorption* jsou zpracovány skrze metodu *sec_sorption*.

- Řádky v oddílu *DualPorosity* jsou zpracovány skrze metodu *sec_dual*.

Po skončení vznikne šest polí obsahující data ze souboru. Tato data jsou pak uložena do výstupního souboru a ten je navrácen modulu *completion.py*.

### 5.7.2 sec_materials (values)

Tato metoda se stará o řádky hodnot oddílu *Materials*. Vstupní parametr je pole hodnot (array). Metoda navrací pole s kompletním záznamem dat (array). První hodnota se uloží jako *material*. Na základě druhé hodnoty se spustí jeden ze sedmi možných zápisů hodnoty *value*. Zápisy se liší počtem parametrů a jejich interpretací.

### 5.7.3 sec_sorption (values)

Tato metoda se stará o řádky hodnot oddílu *Sorption*. Vstupní parametr je pole hodnot (array). Metoda navrací pole s kompletním záznamem dat (array). První hodnota se uloží jako *material*. Druhá hodnota se uloží jako *substance*. Třetí hodnota je převedena na enumerický typ pomocí metody *value_type*. Tato hodnota také ovlivňuje počet prvků v poli *coef*.

### 5.7.4 sec_dual (values)

Tato metoda se stará o řádky hodnot oddílu *DualPorosity*. Vstupní parametr je pole hodnot (array). Metoda navrací pole s kompletním záznamem dat (array). Hodnoty se uloží postupně jako *material*, *mobile*, *immobile* a *coef*. Do coef se uloží zbývající hodnoty v poli.

## 5.8 Testovací moduly

Jejich primárním účelem bylo neustále ověřovat funkčnost jednotlivých metod. Jednalo se o unit-testy, které kontrolovaly, jestli metoda vrací správný výsledek na testované parametry. Tyto unit-testy byly vytvořeny pro většinu metod (některé tímto způsobem testovat nelze). Pomocí těchto testů bylo během tvorby snadnější nalézt případné chyby a také byl neustálý přehled o tom, zda vytvořené metody fungují.

## 5.9 Template soubor

Template soubor je jednoduchý textový soubor obsahující informace pro druhou fázi konverze. K tomuto souboru bude mít přístup pouze správce programu, aby mohl v případě potřeby pozměnit některé parametry. Soubor má přesně definovanou strukturu, aby se předešlo

neočekávaným chybám. Poznámky je možné vkládat vždy na konec řádku bez použití zvláštních znaků (výjimku tvoří prázdné řádky). Každý řádek může mít jeden z následujících formátů:

- Prázdný řádek nebo komentář začínající „//“.
  - Tyto řádky jsou pouze pro informování správce. Metody je nijak nevyužívají.
- [Název oddílu]
  - Určuje, z jaké adresy se mají následující hodnoty hledat. Algoritmus si pamatuje poslední navštívený oddíl.
- Název hodnoty = NULL
  - Tímto se označuje hodnota, která není pro výpočty programu Flow123d nijak důležitá. Takováto hodnota se nikde nenapíše.
  - Případná poznámka nesmí obsahovat znak „=“.
- Název hodnoty = „adresa nové proměnné=datový typ“
  - Toto je nejčastější instrukce. Na novou adresu se uloží hodnota proměnné pozměněná datovým typem.
  - Adresa nové proměnné se píše ve tvaru: „/adresa1/adresa2/název hodnoty“ (adresa může mít samozřejmě i jinou délku)
  - Datový typ může být jeden z následujících:
    - string – řetězec
    - int – celočíselné číslo
    - double – číslo s pohyblivou desetinnou čárkou
    - bool – true/false hodnota
    - data(hodnota) – pevné přiřazení hodnoty
    - enum(1 > hodnota1, 2 > hodnota2, 3 > hodnota3) – výčtový typ. Proměnná je nahrazena jednou z pravých hodnot dvojic podle shody s levou polovinou dvojice.

Poslední část souboru je oddíl *Settings*. Po něm následuje řada možných nastavení pro některé části konverze (například *Decays*). Pro fungování je nutné, aby se jednotlivé řádky nastavení neprohazovaly (zachovalo se jejich pořadí). Nastavení je ve formátu:

Název nastavení = „hodnota“

- Hodnota je řetězec.
- Název nastavení může být libovolný řetězec, vnitřní parser nebere tento název v potaz (rozhoduje pořadí). Název je jen informativní.

```
[Solver]
Use_last_solution = NULL
Keep_solver_files = NULL
Manual_solver_run = NULL
Use_control_file  = NULL
Control_file      = NULL
Solver_accuracy   = "/problem/primary_equation/solver/accuracy=double"
Solver_name       = "/problem/primary_equation/solver/TYPE=string"
Solver_params     = "/problem/primary_equation/solver/parameters=string"
NSchurs     = "/problem/primary_equation/n_schurs=int"
max_it      = "/problem/primary_equation/solver/max_it=int"
POS_format = "/system/output_streams/1/format=enum( ASCII>gmsh_ascii, VTK_SERIAL_ASCII>vtk_ascii)"
```

*Obr. 5: Příklad vzhledu template souboru*

## 5.10 Používání konvertoru

Používání konvertoru je celkově jednoduché. Stačí ho spustit s vyplněnými parametry. Seznam parametrů je uveden v podkapitole *Modul Converter.py*, ale lze ho také vyvolat přidáním parametru „-h“. Ukázka je na následujícím obrázku.

```
converter>python converter.py -h
Program started.
usage: converter.py [-h] [-ini FILE_INI] [-json FILE_JSON] [-final FILE_FINAL]
                    [-sw] [-nc]

Enter parameters for conversion.

optional arguments:
  -h, --help          show this help message and exit
  -ini FILE_INI       Adress or name of source file in ini.
  -json FILE_JSON     Adress or name of source file in json.
  -final FILE_FINAL   Adress or name of converted file.
  -sw                 Show not found messages during conversion.
  -nc                 Disable saving of comments.
```

*Obr. 6: Spuštění nápovědy konvertoru přes příkazový řádek*

Pro úspěšné spuštění je nutné správně vyplnit alespoň první nebo druhý parametr (ini soubor nebo JSON soubor). Na dalším obrázku je ukázán příklad použití konvertoru. Je zvolen zdrojový soubor ve formátu ini a finální soubor ve formátu JSON. Také je spuštěna funkce zobrazování

varování při nenalezení hodnoty ze seznamu. Na výstupu je vidět průběh konverze a zobrazená varování. Název druhého se doplnil automaticky, protože v parametrech chybí.

```
converter>python converter.py -ini flow.ini -final result.json -sw
Program started.
Name of source file in json: flow_partial.json
Conversion to JSON format was sucessful.
Global/Time_step not found.
Global/Density_step not found.
Input/Initial not found.
Input/Sources not found.
Input/sources_formula not found.
Transport/Reactions not found.
Transport/bc_times not found.
Solver/Solver_params not found.
Solver/NSchurs not found.
Solver/max_it not found.
Output/balance_output not found.
Decays will be computed.
Conversion was sucessful.
Program terminated.
```

*Obr. 7: Spuštění konvertoru přes příkazový řádek*

# 6 Vyhodnocení řešení

Zadání práce bylo splněno. Konvertor vstupních dat pro Flow123d pracuje podle očekávání. Všechny konvertované soubory odpovídají určeným normám. Konvertování bylo zkoušeno na několika různých souborech. Díky minimálnímu množství interakce ze strany uživatele se program vyhnul řadě možných chyb a ošetření výjimek se tím značně zjednodušilo.

Možné problémy mohou nastat v případě, že by byla potřeba dlouhá adresa (více než 5 prvků). V tom případě by bylo nutné upravit metodu zodpovědnou za ukládání dat při druhé fázi konvertování (*update_data* v modulu *completion.py*). Dalším problémem může být vytvoření dalších výjimečných událostí pro konvertování. V takovém případě by se musela upravit pravděpodobně celá řada metod v *completion.py.*

## 6.1 Ověření funkčnosti

Pro ověření funkčnosti byl vytvořen jednoduchý soubor ve formátu ini. Na tento soubor se následně spustila konverze. Zdrojový soubor obsahuje většinu možných výjimek a možností. První obrázek ukazuje zdrojový soubor (text je rozdělen do dvou sloupců pro lepší přehlednost):

```
[Global]
Problem_type     = 1
Description        = test2
Stop_time    = 5    //some comment

[Transport]
Transport_on       = Yes
Transport_out_im       = NULL
N_substances         = 2
Substances       = A B C
Substances_density_scales = 1 1 1

[Reaction_module]
Compute_decay = Yes
Nr_of_decay_chains = 2
Nr_of_FoR = 1
```

```
[Decay_1]
Nr_of_isotopes = 3
Substance_ids = 7 3 4
Half_lives = 0.5 0.2

[Decay_2]
Nr_of_isotopes = 4
Substance_ids = 2 1 7 8
Half_lives = 0.5 0.5 0.5
Bifurcation_on = Yes
Bifurcation = 0.2 0.2 0.6

[FoReact_1]
Kinetic_constant = 0.277258872
Substance_ids = 4 5
```

*Obr. 8: Ověření funkčnosti - ini soubor*

Tento zdrojový soubor tedy obsahuje všechny možné datové typy (integer, float, string, boolean, array, null). Na čtvrtém řádku je rovněž komentář (pro konverzi bylo nastaveno zachování komentářů). Soubor také obsahuje všechny výjimky (*Transport_on*, *N_substances*, *New keys* a

*Decays*). Na dalším obrázku je vidět přímá konverze tohoto souboru do formátu JSON. Pro lepší přehlednost je text opět rozdělen do sloupců.

```
{
  "Global" : {
    "Problem_type" : 1,
    "Description" : "test2",
    "Stop_time" : 5,
    "Stop_time_comment" : "some comment"
  },

  "Transport" : {
    "Transport_on" : true,
    "Transport_out_im" : null,
    "N_substances" : 2,
    "Substances" : ["A", "B", "C"],
    "Substances_density_scales" : [1, 1, 1]
  },

  "Reaction_module" : {
    "Compute_decay" : true,
    "Nr_of_decay_chains" : 2,
    "Nr_of_FoR" : 1
  },

  "Decay_1" : {
    "Nr_of_isotopes" : 3,
    "Substance_ids" : [7, 3, 4],
    "Half_lives" : [0.5, 0.2]
  },

  "Decay_2" : {
    "Nr_of_isotopes" : 4,
    "Substance_ids" : [2, 1, 7, 8],
    "Half_lives" : [0.5, 0.5, 0.5],
    "Bifurcation_on" : true,
    "Bifurcation" : [0.2, 0.2, 0.6]
  },

  "FoReact_1" : {
    "Kinetic_constant" : 0.277258872,
    "Substance_ids" : [4, 5]
  }
}
```

*Obr. 9: Ověření funkčnosti - soubor v JSON formátu*

Na obrázku je vidět změna oproti původnímu souboru. Celý soubor je rozdělen do oddílů ohraničených pomocí složených závorek. Jednotlivé názvy oddílů a všechny názvy hodnot jsou ohraničeny pomocí uvozovek. Číselné hodnoty se nijak nezměnily, ale řetězce jsou obohaceny o uvozovky a fráze yes/no jsou nahrazeny booleanovskými hodnotami. Pole jsou teď ohraničená v hranatých závorkách a jednotlivé prvky odděleny čárkou. Čárkou jsou rovněž odděleny všechny oddíly a hodnoty. Komentář je uložen jako další hodnota pod upraveným názvem hodnoty, u které byl uložen (JSON nepodporuje přímo vložené komentáře).

Tento vzniklý soubor se pak následně s pomocí template souboru konvertoval do správné podoby. Díky délce výsledného souboru je zde zobrazen po několika částech a v lehce proházeném pořadí klíčů (za normálních okolností se řadí abecedně):

```
{
  "problem": {
    "TYPE": "sequential_coupling",
    "description": "test2",
    "primary_equation": {
      "TYPE": "steady_mh",
      "output": {
        "pressure_p0": "flow_output_stream",
        "pressure_p1": "flow_output_stream",
        "velocity_p0": "flow_output_stream"
      }
    },
    "secondary_equation": {
      "output": {
        "mobile_p0": "transport_output_stream"
      },
      "substances": ["A", "B"]
    },
    "time_governor": {
      "end_time": 5.0,
      "end_time_comment": "some comment"
    }
  },
```

*Obr. 10: Ověření funkčnosti - finální soubor, část 1*

Na první části je vidět především změna adres u všech zadaných parametrů. Také jsou zde navíc doplněny nové hodnoty z výjimky *New keys*. Pole *Substances* bylo omezeno ze tří prvků na dva díky hodnotě v proměnné *N_substances*. Komentář je vložen pod pozměněným názvem opět hned za hodnotu, za kterou byla v ini souboru uložena. Protože byla hodnota *Transport_on* nastavena na *true*, uložily se ostatní hodnoty z oddílu *Transport* pod nezměněnými názvy (*Transport_on* samotný se neuložil). Následuje druhá část výsledného souboru:

```
  "system": {
    "output_streams": [
      {
        "name": "flow_output_stream"
      },
      {
        "name": "transport_output_stream"
      }
    ]
  },
  "zz_not_supported": {
    "transport": {
      "substances_density_scales": [1, 1, 1]
    }
  },
```

*Obr. 11: Ověření funkčnosti - finální soubor, část 2*

Tato část ukazuje dva další prvky. Prvním prvkem je použití pole skupin hodnot. Hodnota *output_streams* obsahuje jako svoji hodnotu pole, kde každým prvkem je samostatný oddíl obsahující další hodnoty. Druhým prvkem je automatické ukládání nevyužívaných hodnot na konec celého souboru (zde je uprostřed jen pro přehlednost, normálně je na konci). Po této části následuje zpracování výjimky *Decays*:

V této části je vidět zpracování *Decays* (jednotlivá pole byly upraveny tak, aby se vešly na jednu řádku). Z výpisu je vidět, že se první oddíl *Decay_1* rozdělil na dva záznamy navazující na sebe. *Decay_2* se uložil společně s polem *Probability* (protože hodnota *Bifurcation_on* byla *true*). Oddíl *FoReact_1* se uložil jako další oddíl, akorát s jiným názvem hodnoty (*Half_life* změněno na *Kinetic*).

Konverze tedy byla úspěšná a výsledný soubor odpovídá definovaným pravidlům. Konvertor je tedy připraven na jeho používání uživateli a tvůrci programu Flow123d.

```
"reactions": {
  "decays": [
    {
      "Half_life": 0.5,
      "Substances": [7, 3]
    },
    {
      "Half_life": 0.2,
      "Substances": [3, 4]
    },
    {
      "Half_life": 0.5,
      "Probability": [0.2, 0.2, 0.6],
      "Substances": [2, 1, 7, 8]
    },
    {
      "Kinetic": 0.277258872,
      "Substances": [4, 5]
    }
  ]
}
}
```

*Obr. 12: Ověření funkčnosti - finální soubor, část 3*

# 7 Závěr

## 7.1 Cíle práce

Po konvertoru byla požadována celá řada vlastností. Jako celek měl být plně obslužitelný z příkazové řádky. Také po něm bylo požadováno, aby dokázal kromě samotné konverze i výsledný soubor upravit (změna adres hodnot, změna datového typu, atd.). Pro konverzi také bylo stanoveno několik výjimek, které můžou ovlivnit proces konverze. Všechny změny by měly být upravitelné přes template soubor. Program by také měl uložit data ze souboru s materiály do vzniklého souboru.

## 7.2 Návrh programu

Jako programovací jazyk pro tento konvertor byl zvolen Python pro svoji univerzálnost a přenositelnost. Python navíc obsahuje standardně knihovnu *json*, která byla pro konverzi potřeba. Samotná konverze byla rozdělena do tří fází. První fáze byla převedení ini formátu na JSON. Druhá fáze byla upravení vzniklého souboru podle instrukcí uložených na template souboru. Třetí je zpracování souboru s materiály. Pro každou fázi byl určen samostatný modul.

## 7.3 Popis řešení

Konvertor na základě vložených parametrů otevře ini soubor a přeloží ho do JSON formátu. Jednotlivé hodnoty a klíče jsou upraveny do požadovaného tvaru. Následně se vzniklý JSON soubor upravuje podle jednotlivých instrukcí v template souboru včetně výjimek. Poté se otevře soubor s materiály a výsledná data se uloží do souboru. Text se pak upravuje pro lepší čitelnost.

## 7.4 Splnění cílů

Všechny cíle práce byly splněny.

Konvertor byl několikrát důkladně otestován (více v kapitole 6.1 Ověření funkčnosti). Byly použity soubory, které obsahovaly všechny možné výjimky a hodnoty. U současné verze nebyly nalezeny žádné chyby ve výsledných souborech – výsledné soubory odpovídají tvaru, jaký byl uveden v dokumentaci Flow123d. Soubor s materiály se také konvertuje podle požadavků. Jediný známý problém je omezení velikosti nových adres, ačkoliv současné verze template souboru takto dlouhou adresu nevyužívají.

## 7.5 Budoucnost konvertoru

Budoucnost konvertoru záleží na tom, jakým směrem se vydá vývoj programu Flow123d. Na základě požadavků můžou být do konvertoru zabudovány nové výjimky, nové vstupní parametry nebo se můžou upravit stávající omezení.

# Seznam použité literatury


- PYTHON SOFTWARE FOUNDATION. *Python*: *About* [online]. 1990 [cit. 2012-05-10]. Dostupné z: http://www.python.org/about/
- CHEEVER, Charlie. Which Python JSON library should I use. *Quora* [online]. 2010 [cit. 2012-05-10]. Dostupné z: http://www.quora.com/Which-Python-JSON-library-should-I-use
- *JSON*: *Introducing JSON* [online]. 1999 [cit. 2012-05-10]. Dostupné z: http://www.json.org/
- BŘEZINA, Jan. *FLOW123D*: *draft, scheme of new inputs*. Liberec, 2011.
- PILGRIM, Mark. *Dive into Python* [online]. 2004 [cit. 2012-05-10]. Dostupné z: http://www.diveintopython.net/toc/index.html