Zadání projektu z předmětu IPP 2009/2010

Zbyněk Křivka a Dušan Kolář
E-mail: {krivka, kolar}@fit.vutbr.cz, {54 114 1313, 54 114 1238}

1 Základní charakteristika projektu

Název projektu:

Návrh, implementace a dokumentace dvou skriptů v jazyce Perl a Python

Revize zadání: Jako každý dokument i toto zadání bude mít jistě pár chyb, a tudíž prosím o pomoc jich co nejvíce odhalit do konce února, kdy se stane zadání naprosto definitivní a neměnné. V žádném případě nepůjde o změny zásadní, ale spíše kosmetické, takže můžete začít okamžitě pracovat na řešení projektu.

2010-02-15: Vznik zadání a vytvoření konkrétních zadání úloh

2010-02-23: Zveřejnění zadání, registrace bude spuštěna koncem týdne

2010-04-08: Oprava překlepu u hlavičky XML a několika dalších překlepů.

Projekt je individuální a sestává se z naprogramování skriptů pro **dvě úlohy**, které si student vybírá z možných variant (viz sekce 2). První skript bude povinně implementován v jazyce **Perl** a druhý v jazyce **Python** (doporučujeme použití technik OOP v kombinaci s dalšími programovacími paradigmaty).

Skripty jsou aplikace příkazové řádky (tzv. filtry) se společným základem zadání (viz sekce 3). Každá úloha (viz sekce 4) má identifikátor ze tří písmen (např. CST, CSV, JSN, SMS, XQR).

2 Odevzdání

Forma odevzdání: Projekt se odevzdává individuálně prostřednictvím IS FIT v předmětu IPP (odevzdání emailem není možné, a to ani po termínu). Všechny odevzdané soubory k dané úloze budou zkomprimovány programem TAR+GZIP či ZIP do jediného archivu pojmenovaném `xlogin00-XYZ.zip` nebo `xlogin00-XYZ.tgz`, kde `xlogin00` je Váš login a `XYZ` je identifikátor úlohy implementované a dokumentované v daném archívu. Velikost každého archívu bude omezena informačním systémem (pravděpodobně na 1 MB). Archív nesmí obsahovat speciální či spustitelné soubory. Názvy všech souborů mohou obsahovat pouze písmena anglické abecedy, číslice, tečku, pomlčku a podtržítko. Archív nebude obsahovat žádné adresáře výjma adresářů pro přiložené povolené knihovny.

V rámci prvního termínu se ve WIS přihlásíte na variantu (dvě úlohy s určením konkrétních implementačních jazyků). Do prvního termínu budete také odevzdávat archív pro první úlohu v jazyce Perl. Několik dní po uzavření prvního termínu bude otevřen druhý termín pro odevzdání archívu pro druhou úlohu v jazyce Python.

Datum odevzdání:

Termíny: první skript **9. dubna** 2010 do 23:59:59, druhý skript **30. dubna** 2010 do 23:59:59

Dodatečné informace a konzultace (k obsahu zadání pro IPP):

– Často kladené otázky k projektu z předmětu IPP,
viz `http://www.fit.vutbr.cz/study/courses/IPP/private/faq.html`
– Fórum předmětu IPP pro ak. rok 2009/2010, témata `IPP: Projekt.*`
– Zbyněk Křivka: po dohodě e-mailem (uvádějte předmět začínající `"IPP:"`), viz `http://www.fit.vutbr.cz/~krivka`
– Dušan Kolář: po dohodě e-mailem (uvádějte předmět začínající `"IPP:"`), viz `http://www.fit.vutbr.cz/~kolar` (jen v závažných případech)

Hodnocení:

Výše bodového hodnocení projektu v předmětu IPP je **celkem 20 bodů**.

Každý skript včetně dokumentace bude hodnocen **10 body**. Z toho bude dokumentace až za 2 body, avšak maximálně 30% hodnocení skriptu (tedy v případě neodevzdání funkčního skriptu bude samotná dokumentace hodnocena 0 body).

Za řešení je možné získat až 25% bonusových bodů, kdy zisk se počítá z aktuálního bodového zisku (např. při získání 20 bodů může být bonus až 5 bodů, při získání 16 bodů může být bonus až 4 body, hranice se zaokrouhluje nahoru). Takový bonus je možné získat pouze za obzvláště kvalitní a podařené řešení některého z registrovaných rozšíření (viz zadání jednotlivých úloh v sekci 4), kvalitativně nadprůměrnou účast na fóru projektu apod.

Skripty budou spouštěny příkazem: *interpret skript parametry*, kde *interpret* bude `perl` nebo `python`, jméno *skript* závisí na dané úloze.

Způsob odevzdání: Skript s dokumentací budou součástí řešení každé úlohy. Každá úloha bude odevzdána ve zvláštním archívu (viz forma odevzdání, která je popsána na začátku této sekce). Skript bude umístěn v kořenovém adresáři odevzdávaného systému souborů a odtud se bude i spouštět.

Pokud tyto pokyny nebudou dodrženy, projekt se považuje za neodevzdaný! Dále je třeba odevzdat zadanou část projektu v daném termínu (viz výše). Pokud tomu tak nebude, je projekt opět považován za neodevzdaný, nebo minimálně se dají očekávat významné bodové srážky (cca bod za každou započatou hodinu zpoždění). Stejně tak pokud se bude jednat o plagiátorství jakéhokoliv druhu, je projekt ohodnocen 0 body, nebude udělen zápočet a bude zvážen další postup.

Projekt je hodnocen jako neodevzdaný (shrnutí), pokud:

- nebude doručen včas
- nebude doručen v předepsaném formátu
- nebude možné interpretovat skript tak, jak je uvedeno v zadání (např. je nutný další zásah do skriptu, ruční nastavení přístupových cest, apod.)
- půjde o plagiát v jakékoliv podobě
- ...

Pokud student nezachová kterékoliv z výše uvedených pravidel, zadavatelé projektu mají právo jej hodnotit nulou.

Pokud máte jakékoliv dotazy, problémy, či nejasnosti ohledně tohoto projektu, tak po přečtění často kladených otázek a fóra předmětu, neváhejte napsat na fórum (u obecného problému, jež se potencionálně týká i vašich kolegů) či kontaktovat garanta projektu (v případě individuálního problému, e-mail: krivka@fit.vutbr.cz) nebo nouzově garanta předmětu (e-mail: kolar@fit.vutbr.cz), kdy do předmětu uveďte na začátek řetězec `"IPP:"`*. Na problémy zjištěné v řádu hodin až jednotek dní před termínem odevzdání nebude brán zřetel, začněte proto řešit s dostatečným předstihem.*

Pokud se při vytváření projektu dopustíte jakéhokoliv prohřešku proti akademické morálce (např. plagiátorství), váš přestupek bude ohlášen garantům příslušných předmětů a může vyústit k vyloučení z akademické obce a tím pádem celého VUT.

Varianty projektu: Projekt se skládá z několika úloh, z nichž si student vybírá dvojici úloh (první implementuje v Perlu, druhou v Pythonu) z následujících povolených variant:

1. CST+CSV (C Stats v Perlu, CSV2XML v Pythonu) [viz sekce 4.2 a 4.4]
2. CST+JSN (C Stats v Perlu, JSON2XML v Pythonu) [viz sekce 4.2 a 4.5]
3. CST+SMS (C Stats v Perlu, SMS Compress v Pythonu) [viz sekce 4.2 a 4.3]
4. CSV+CST (CSV2XML v Perlu, C Stats v Pythonu) [viz sekce 4.4 a 4.2]
5. CSV+XQR (CSV2XML v Perlu, XML Query v Pythonu) [viz sekce 4.4 a 4.1]
6. JSN+CST (JSON2XML v Perlu, C Stats v Pythonu) [viz sekce 4.5 a 4.2]
7. JSN+XQR (JSON2XML v Perlu, XML Query v Pythonu) [viz sekce 4.5 a 4.1]
8. SMS+CST (SMS Compress v Perlu, C Stats v Pythonu) [viz sekce 4.3 a 4.2]
9. SMS+XQR (SMS Compress v Perlu, XML Query v Pythonu) [viz sekce 4.3 a 4.1]
10. XQR+CSV (XML Query v Perlu, CSV2XML v Pythonu) [viz sekce 4.1 a 4.4]
11. XQR+JSN (XML Query v Perlu, JSON2XML v Pythonu) [viz sekce 4.1 a 4.5]
12. XQR+SMS (XML Query v Perlu, SMS Compress v Pythonu) [viz sekce 4.1 a 4.3]

3 Společný základ zadání všech úloh

1. Implementujte v daném programovacím jazyce programové vybavení (skript) pro řešení přidělené úlohy.
2. Toto programové vybavení doplňte o vhodnou implementační dokumentaci.

3.1 Dokumentace

Implementační dokumentace (dále jen dokumentace) musí být stručným a uceleným průvodcem **vašeho způsobu řešení** zadané úlohy. Bude vytvořena ve formátu **PDF**. Jakékoliv jiné formáty dokumentace než PDF budou ignorovány, což povede ke ztrátě bodů za dokumentaci. Dokumentaci je možné psát česky (s diakritikou, formálně čistě) anebo anglicky (formálně čistě). Dokumentace může být doplněna o graf pro použitý konečný automat, pravidla Vámi vytvořené gramatiky nebo popis jiný speciálních technik a algoritmů. Nicméně **nesmí obsahovat ani částečnou kopii zadání**.

Dokumentace bude dále popisovat způsob a postup řešení a celkovou filozofii návrhu. **Rozsah** textu je minimálně jedna a maximálně dvě strany A4 (10-bodové písmo Times New Roman a případně Courier pro indetifikátory a skutečně krátké úryvky kódu; vynechávejte zvláštní úvodní stranu, obsah i závěr) pro řešení každé úlohy.

Nadpis a hlavička dokumentace bude na prvních čtyřech řádcích obsahovat:

```
Dokumentace úlohy %XYZ%: %task% v %language% do IPP 2009/2010
Jméno a příjmení: %name_sirname%
Login: %xlogin00%
Varianta projektu do IPP: %variant%
```

kde `%XYZ%` je identifikátor vaší úlohy, `%task%` jméno vaší úlohy, `%language%` implementační jazyk dané úlohy (Perl nebo Python), `%name_sirname%` vaše jméno a příjmení, `%xlogin00%` váš login a nakonec `%variant%` zastupuje zkrácené označení varianty projektu.

Dokumentace bude v hlavním adresáři odevzdaných souborů a pojmenována `XYZ-doc.pdf`, kde `XYZ` je identifikátor úlohy.

V rámci dokumentace bude hodnoceno i komentování zdrojového kódu skriptu (minimálně každá funkce a modul by měly mít svůj komentář o jejich funkci; u složitějších funkcí dokumentujte i omezení na vstupy či výstupy). Každý skript (i případný pomocný) musí v řádkovém komentáři na druhém[1] řádku zdrojového textu obsahovat dvojtečkou oddělený identifikátor úlohy, název úlohy, login a celé jméno autora skriptu. Například (v Perlu i Pythonu):

```
#XQR:XML Query:xnovak99:Jan Novak
```

3.2 Programová část

Zadání úloh vyžadují implementaci textového filtru nebo jiné aplikace příkazové řádky[2], která má vstupní parametry a je definováno, jakým způsobem manipuluje se vstupy a výstupy. Veškeré chybové hlášky, varování a ladicí výpisy směřujte pouze na standardní chybový výstup, jinak pravděpodobně nedodržíte zadání kvůli modifikaci definovaných výstupů (ať již do externích souborů nebo do standardního výstupu). Jestliže proběhne činnost skriptu bez chyb, vrací se výstupní návratová hodnota `0` (nula). Jestliže došlo k nějaké chybě (špatné parametry, neexistující soubor, chybná syntaxe), vrací se výstupní návratová hodnota větší jak nula.

V případě, že vstupem je textový soubor, CSV soubor, JSON soubor nebo XML soubor, tak uvažujte, že je v kódování UTF-8. V případě XML souborů není třeba uvažovat deklarace jmenných prostorů. Všechna vstupní XML budou v tomto smyslu nekolizní a nebudou jmenné prostory využívat. Nicméně deklarace se v XML souboru vyskytnout může a tu potom ignorujte.

[1] Tato informace může být v Pythonu na třetím řádku, pokud druhý bude obsahovat informaci o kódování zdrojového textu.

[2] konzolové aplikace

Každý skript se bude jmenovat podle identifikátoru úlohy převedeného na malá písmena a příponu[3] bude mít podle daného skriptovacího jazyka dle platných zvyklostí (Perl (`.pl`), Python (`.py`) nebo Python 3 (`.py3`)). Vyhodnocení skriptů bude prováděno na serveru `Merlin` s aktuálními verzemi interpretů (dne 16. 2. 2010 byl na tomto serveru nainstalován `perl` verze 5.8.8, `python` verze 2.5.4 nebo `python3.0` verze 3.0.1).

K řešení lze využít standardně předinstalované knihovny obou jazykových prostředí na serveru `Merlin`. V případě využití jiné knihovny kromě knihovny podporující načítání formátů XML, popř. CSV či JSON je třeba konzultovat s garantem projektu (především z důvodu, aby se řešení projektu použitím vhodné knihovny nestalo zcela triviálním).

Každý skript bude pracovat s těmito společnými parametry:

- `--help` vypíše na standardní výstup nápovědu skriptu, kterou lze převzít z tohoto zadání (lze odstranit diakritiku, případně přeložit do angličtiny dle zvoleného jazyka dokumentace). Tento parametr nelze kombinovat s žádným dalším parametrem, jinak skript ukončete s chybou.
- `--input=filename.ext` zadaný vstupní soubor může být zadán relativní cestou nebo absolutní cestou; v případě, že by název či cesta obsahovala mezeru nebo uvozovky, tak musí být uvedena celá cesta i se jménem souboru v uvozovkách (uvozovky ve jméně souboru není třeba uvažovat); chybí-li tento parametr, tak je uvažován standardní vstup.
- `--output=filename.ext` zadaný výstupní soubor může být zadán relativní cestou nebo absolutní cestou; v případě, že by název či cesta obsahovala mezeru, tak musí být uvedena cesta i se jménem souboru v uvozovkách (uvozovky ve jméně souboru není třeba uvažovat); chybí-li tento parametr, tak je výstup přesměrován na standardní výstup.

Kombinovatelné parametry skriptů mohou být uváděny v libovolném pořadí, pokud nebude řečeno jinak. Zakázané kombinace parametrů nebo použití neznámého parametru ukončí skript s chybou.

Reference:

- W3C, Extensible Markup Language (XML) 1.0 (5. vydání), 2008. Dostupné na `http://www.w3.org/TR/REC-xml/ (citováno 22. 2. 2010)`

4 Zadání jednotlivých úloh

Následují podrobné zadání jednotlivých úloh, které po dvojicích tvoří možné varianty projektu.

4.1 XQR: XML Query

Skript provádí vyhodnocení zadaného dotazu, jenž je podobný příkazu SELECT jazyka SQL, nad vstupem ve formátu XML. Výstupem je XML obsahující elementy splňující požadavky dané dotazem. Dotazovací jazyk má zjednodušené podmínky a syntaxi.

Tento skript bude pracovat s těmito parametry:

- `--help` viz výše
- `--input=filename.ext` zadaný vstupní soubor ve formátu XML

[3]Pomocné skripty nebo knihovny mohou mít jinou příponu. Např. `.py` pro Python 3 nebo `.pm` pro Perl.

- `--output=filename.ext` zadaný výstupní soubor ve formátu XML s obsahem podle zadaného dotazu
- `-q='dotaz'` zadaný dotaz v dotazovacím jazyce definovaném níže (v případě zadání tímto způsobem nebude dotaz obsahovat symbol apostrof)
- `-qf=filename.ext` dotaz v dotazovacím jazyce definovaném níže zadaný v externím textovém souboru (nelze kombinovat s `-q`)
- `-n` negenerovat XML hlavičku na výstup skriptu
- `-r=element` jméno párového kořenového elementu obalující výsledky. Pokud nebude zadán, tak se výsledky neobalují kořenovým elementem, ač to porušuje validitu XML.

Dotazovací jazyk Nejprve uveďme neformální zápis struktury dotazovacího jazyka:

```
SELECT element LIMIT n FROM element|ROOT WHERE condition
                       ORDER BY element|attribute ASC|DESC
```

Celá bezkontextová gramatika (včetně povolených rozšíření) je definována takto (neterminály jsou v úhlových závorkách, `<QUERY>` je startující netermínál, tokeny jsou odděleny bílým znakem a jazyk je case-sensitive[4]):

```
<QUERY> --> SELECT element <LIMITn> FROM <FROM-ELM> <WHERE-CLAUSE> <ORDER-CLAUSE>
<LIMITn> --> empty
<LIMITn> --> LIMIT number
<FROM-ELM> --> element
<FROM-ELM> --> ROOT
<WHERE-CLAUSE> --> empty
<WHERE-CLAUSE> --> WHERE <CONDITION>
<CONDITION> --> ( <CONDITION> )
<CONDITION> --> NOT <CONDITION>
<CONDITION> --> <CONDITION> AND <CONDITION>
<CONDITION> --> <CONDITION> OR <CONDITION>
<CONDITION> --> <ELEMENT-OR-ATTRIBUTE> <RELATION-OPERATOR> <LITERAL>
<LITERAL> --> string
<LITERAL> --> number
<RELATION-OPERATOR> --> CONTAINS
<RELATION-OPERATOR> --> =
<RELATION-OPERATOR> --> >
<RELATION-OPERATOR> --> <
<ELEMENT-OR-ATTRIBUTE> --> element
<ELEMENT-OR-ATTRIBUTE> --> attribute
<ORDER-CLAUSE> --> empty
<ORDER-CLAUSE> --> ORDER BY <ELEMENT-OR-ATTRIBUTE> <ORDERING>
<ORDERING> --> ASC
<ORDERING> --> DESC
```

Implementace podpory klauzule `ORDER BY` je za bonusový bod. Podpora skládání podmínek pomocí závorek a klíčových slov `AND`, `OR` a `NOT` je za bonusové 2 body. Nejvyšší prioritu má operátor `NOT`, nižší má `AND` a nejnižší `OR`. Operátory `AND` a `OR` jsou levě asociativní.

[4] *case-sensitive* znamená, že v dotazech záleží na velikosti písmen u klíčových slov i u identifikátorů.

Lexémy jsou definovány takto: `empty` je prázdný řetězec. `number` je celé číslo v běžném 32-bitovém celočíselném rozsahu implementačního jazyka. `element` je identifikátor elementu jazyka XML (bez ohraničujících zobáčků). `attribute` je identifikátor atributu se znakem zavináč (`@`) na začátku jako prefix. `string` je řetězec zapsaný v uvozovkách, který neobsahuje žádné netisknutelné znaky, escape sekvence, konec řádku, ani uvozovky (nebude testováno).

Sémantika dotazovacího jazyka: Dotaz v klauzuli `FROM` definuje zdrojový element (viz neterminál `<FROM-ELM>`), kde se následně hledají vnořené výstupní elementy z klauzule `SELECT`, které splňují podmínky dané v klauzulí `WHERE`. Poté může být výsledný seznam elementů seřazen klauzulí `ORDER BY` a ořezán omezením `LIMIT` na požadovaný maximální počet elementů.

Hledání zdrojového elementu provádějte hledáním do hloubky, dokud nenarazíte na první výskyt zdrojového elementu, kde bude prováděno další zpracování (již se neuvažuje další nepřekrývající zdrojový element jinde na vstupu). Vzájemné zanoření totožných výstupních elementů neřešte. Nicméně uvažujte, že výstupní element může být pokaždé zanořen do jiné úrovně ve zdrojovém elementu (nepřekrývajícím způsobem). Klíčové slovo `ROOT` zastupuje virtuální kořenový element zastupující celý XML dokument, který pak obsahuje skutečný kořenový element. Entita (element nebo atribut) z podmínky v klauzuli `WHERE` se hledá opět do hloubky po první svůj výskyt. Není-li tato entita v aktuálně kontrolovaném výstupním elementu nikde nalezena nebo je její první výskyt špatného typu, je výsledek porovnání (viz neterminál `<RELATION-OPERATOR>`) nepravdivý.

Výstupní elementy jsou na výstup kopírovány v nezměněné podobě (včetně všech atributů, hodnot i podelementů[5]), případně obaleny kořenovým elementem v závislosti na parametrech skriptu.

V případě kolize jmen atributů či elementů uvažujte první načtený.

Příklady k definici sémantiky dotazovacího jazyka:

```
SELECT book FROM library WHERE title CONTAINS "XML"
```

Projdi všechny elementy `<book>` v elementu `<library>` a vyber ty knihy, které v prvním podelementu `<title>` obsahují podřetězec `XML` (case-sensitive[6]). Pokud `<title>` obsahuje další elementy místo textové hodnoty, tak skonči s chybou. Vybrané knihy vypisujte na výstup jako kopie vybraných elementů `<book>` včetně všech atributů a podelementů (až na formátování). Podle argumentů skriptu případně doplňte párový kořenový element a XML hlavičku.

```
SELECT book LIMIT 8 FROM library WHERE title CONTAINS "Duna" AND
                                 (@name = "Frank Herbert" OR year > 1980)
                          ORDER BY year DESC
```

Vypište XML elementy (vč. podelementů a atributů) pro 8 knih, jejichž název obsahuje podřetězec `Duna` a jejichž autor uvedený v libovolném prvně načteném atributu `name` uvnitř libovolného elementu uvnitř elementu `<book>` (v libovolném zanoření, tedy i přímo v elementu `<book>`) je `Frank Herbert` nebo rok (element `<year>`) je větší jak číslo `1980`. Výsledek seřaďte sestupně podle obsahu elementu `<year>` a až poté vyber prvních 8 knih (dáno klauzulí `LIMIT`). Pokud dojde ke kolizi jmen například atributu `name` ve více elementech, tak uvažujte první nalezený atribut v elementu `<book>` nebo jeho podelementech. Relační operátor pracující s neexistujícím (nedefinovaným) elementem nebo atributem vždy vrací nepravdivou hodnotu.

```
SELECT title FROM library WHERE NOT title CONTAINS "Duna"
```

[5] Není třeba kopírovat komentáře.

[6] *case-sensitive* vyhledávání řetězců vyžaduje shodu i ve velikosti všech písmen.

Tento dotaz vypíše pomocí XML všechny názvy knih z knihovny, které neobsahují v názvu řetězec `Duna`.

```
SELECT author FROM library WHERE (year > 1980 OR year = 1980) AND year < 1990
```

Poslední příklad vypíše autory z knihovny, kteří publikovali v letech 1980 až 1989 včetně. Na celočíselný literál nelze aplikovat relační operátor `CONTAINS`.

4.2 CST: C Stats

Vytvořte skript pro analýzu zdrojových souborů jazyka C podle standardu C99 (přípona `.c` a `.h`), který vypíše statistiky komentářů, klíčových slov, operátorů a řetězců.

Tento skript bude pracovat s těmito parametry:

- `--help` viz výše
- `--input=fileordir.ext` zadaný vstupní soubor nebo adresář se zdrojovým kódem v jazyce C. Uvažujte soubory v normálním kódování ASCII, ISO 8859-2 nebo CP1250 (nikoli UTF-8). Jeli zadán adresář, tak jsou postupně analyzovány všechny soubory s příponou jazyka C (`.c`, `.h`) v tomto adresáři a jeho podadresářích. Pokud nebude tento parametr zadán, tak se analyzují soubory (opět pouze s příponou `.c` a `.h`) z aktuálního adresáře a všech jeho podadresářů.
- `--output=filename.ext` zadaný textový výstupní soubor v ASCII kódování (přesný formát viz níže)
- `-k` vypíše počet všech výskytů klíčových slov (vyskytujících se mimo poznámky a řetězce) v každém zdrojovém souboru a celkem (pro všechny analyzované soubory)
- `-o` vypíše počet výskytů jednoduchých operátorů (nikoliv oddělovačů apod.) mimo poznámky a řetězce v každém zdrojovém souboru a celkem (pro všechny analyzované soubory). Jednoduchý operátor definujme jako danou dopředu známou a fixní posloupnost nepísmenných znaků[7]. Nepovinné rozšíření zadání: Přesná (v rámci možností) statistika nejednoduchých operátorů se bude započítávat nepovinným parametrem `-s` v kombinaci s `-o` a bude za až 2,5 bonusového bodu. Např. Ternární operátor `?:` je považován za jeden (až 1 bod). Pozor na lexém čárka vystupující někdy jako operátor a jindy jako separátor (až 1 bod). Správné rozpoznání netriviálních použití operátorů indexace, volání funkce a přetypování (až 2 body).
- `-ik` vypíše počet výskytů všech identifikátorů (mimo poznámky a řetězce) v každém zdrojovém souboru a celkem (pro všechny analyzované soubory) — zahrnuje klíčová slova
- `-i` vypíše počet výskytů identifikátorů (mimo poznámky a řetězce) v každém zdrojovém souboru a celkem (pro všechny analyzované soubory) — nezahrnuje klíčová slova
- `-w=<pattern>` vyhledá přesný textový řetězec `pattern` ve všech zdrojových kódech a vypíše počet výskytů na soubor i celkem. Jelikož se nejedná o identifikátor ale řetězec, hledá se i v poznámkách a řetězcových literálech, zatímco v předchozích případech se poznámky a řetězcové literály vynechávaly!)

[7] Jednoduchý operátor tedy není např. `sizeof`, operátor přetypování, indexace nebo volání funkce.

- `-c` vypíše celkový počet znaků komentářů včetně uvozujících znaků komentářů (`//`, `/*` a `*/`) na soubor a celkem (pro všechny analyzované soubory). Komentáře počítejte včetně znaku konce řádku[8] uvnitř blokového komentáře a v případě řádkového komentáře také započítejte znak konce řádku[8] do počtu znaků komentáře.

- `-p` v kombinaci s předchozími (až na `--help`) způsobí, že soubory se budou vypisovat bez úplné cesty k souboru (tedy pouze jména souborů; řazení souborů pak také neuvažuje absolutní cesty k souborům).

Parametry `-k`, `-o`, `-ik`, `-i`, `-w` a `-c` nelze mezi sebou kombinovat a pokud nebude uveden `--help`, tak je požadováno uvedení právě jednoho z těchto parametrů. Formát výpisu bude takovýto (vypisované záznamy souborů jsou seřazeny vzestupně):

```
<jméno souboru příp. i s absolutni cestou>  <správné odsazení>  <číslo>
<jméno souboru příp. i s absolutni cestou>  <správné odsazení>  <číslo>
<jméno souboru příp. i s absolutni cestou>  <správné odsazení>  <číslo>
...                                         ...                ...
CELKEM:                                     <správné odsazení>  <číslo>
```

Přitom `<správné odsazení>` je takový přesný minimální počet mezer, aby všechna čísla končila na stejném sloupci (první sloupec je zarovnán doleva a poslední sloupec doprava) a žádné nezačínalo ve sloupci začínajícím za tím sloupcem[9], kde se, byť na jiném řádku výpisu, vyskytuje text jména souboru. Seznam souborů bude vzestupně seřazen podle abecedy (neuvažujte diakritiku v názvech souborů).

Skript bude prohledávat všechny soubory od místa uložení hlouběji (myšleno k podadresářům), které obsahují soubor s platným rozšířením (pro jazyk C to budou `.c` a `.h`). POZOR! Testovací adresářová struktura se zdrojovými texty jazyka C, ale případně i jinými soubory, které je třeba ignorovat, bude ke skriptu nahrávána námi automaticky.

Makra preprocesoru jazyka C ignorujte (např. vypuštěním před samotným zpracováním zdrojového textu), což znamená, že definice maker budou vynechána podobně jako komentáře (Nezapočítávají se však jako komentáře!). Volání maker ve zbylém zdrojovém kódu nijak speciálně neřešte (většinou se volání makra rozpozná jako identifikátor, což je lexikálně v pořádku). Pozor na fakt, že definice makra může pokračovat i na následujícím řádku při použití znaku \ na konci řádku s definicí makra.

Reference:

- Standard ISO/IEC 9899:TC2 Committee Draft — May 6, 2005. Dostupné na `http://www.open-std.org/JTC1/SC22/wg14/www/docs/n1124.pdf` [Citováno 22. 2. 2010]

4.3 SMS: SMS Compress

Vytvořte skript pro kompresi zadané krátké textové zprávy (SMS), odstranění české diakritiky a případnou aplikaci pravidel ze slovníku zkratek.

[8] Započítávejte `\n` stejně jako `\r\n` obojí jako jeden znak.

[9] Kromě nekolize sloupců je třeba vložit také minimálně jednu mezeru.

Tento skript bude pracovat s těmito parametry:

- `--help` viz výše
- `--input=filename.ext` zadaný vstupní textový soubor s krátkou textovou zprávou
- `--output=filename.ext` textový výstupní soubor v ASCII kódování nebo v UTF-8 v případě, že ve výstupu zůstala diakritika (např. neprovedením jejího odstranění nebo kvůli následné aplikaci nevhodného slovníku).
- `-r` provede odstranění české diakritiky a nahrazení stejnými znaky bez diakritického znaménka. V případě kombinace s `-c` nebo `-v` se provede odstranění diakritiky jako první.
- `-c` provede kompresi SMS podle dodatečných nastavení typu Camel, tj. převod každého slova[10] na Camel notaci[11] a vynechání přebytečných mezer (první písmeno slova je velké, všechna ostatní jsou malá).
- `-a` v kombinaci s `-c` bude na Camel notaci převádět pouze slova[10] ze samých malých písmen. Pokud je tedy část SMS napsána již v Camel notaci, tak ze slova "ZeSlovaTakovehoto" nevznikne slovo "Zeslovatakovehoto", ale zůstane zachováno "ZeSlovaTakovehoto".[12]
- `-b` v kombinaci s `-c` (nepovoleno kombinovat s `-a`) vynechá z komprese slova[10] napsaná velkými písmeny (např. zkratky). Mezery před a za těmito slovy nezachovávejte.
- `-d=filename` určení XML slovníku zkratek (obsahuje pravidla pro zkracování a expanzi zkratek, viz níže), pro jméno souboru platí stejná pravidla jako pro zadávání vstupního či výstupního souboru, implicitní hodnota však neexistuje. Tento parametr vyžaduje uvedení parametru `-v`.
- `-v` provede aplikaci pravidel ze zadaného slovníku zkratek (vyžadováno určení parametrem `-d`, jinak chyba). Aplikace se provádí před samotnou kompresí, pokud je komprese parametrem `-c` vyžadována. Nejprve se provede expanze a poté zkracování. Jeli zadán parametr `-e` nebo `-s`, tak se může provést jenom jedna z aplikací slovníku zkratek (tj. buď expanzivní, nebo zkracující pravidla).
- `-e` aplikuje pouze expanzivní pravidla ze zadaného slovníku zkratek (vyžaduje parametr `-v`; nelze kombinovat s `-s`).
- `-s` aplikuje pouze zkracující pravidla ze zadaného slovníku zkratek (vyžaduje parametr `-v`; nelze kombinovat s `-e`).
- `-n` vypíše minimální počet SMS, na které je nutno výstupní SMS rozdělit. Uvažujme, že jedna SMS může mít 160 znaků bez diakritiky či 70 znaků s diakritikou, ale v případě rozdělení na více zpráv již pouze 153 znaků bez diakritiky nebo 67 znaků s diakritikou na každou zprávu. Výstupem bude v tomto případě pouze číslo (počet SMS) a převedená SMS bude zahozena. Výpočet počtu rozdělených zpráv se provádí na základě výstupu po odstranění diakritiky, po aplikaci pravidel a po komprimaci (je-li to vyžádáno odpovídajícími parametry).

[10] *Slovo* je neprázdný řetězec složený pouze z písmen z obou stran oddělený nepísmenným znakem, tj. bez mezer, interpunkce, číslic a jiných znaků.

[11] Např. "Testovací text" převede na "TestovacíText" nebo "test, JAK38 jinak" převede na "Test,Jak38Jinak".

[12] Špatné použití Camel notace, jako např. začátek malým písmenem, neřešte.

Slovník zkratek ve formátu XML je v kódování UTF-8. Po XML hlavičce `<?xml version="1.0" encoding="UTF-8"?>` následuje kořenová značka `<sms-abbreviation-dictionary>`, která obsahuje pravidla `<rule>` (párový element) s atributy `expansive="1"` (v případě expanzivního pravidla; jinak je pravidlo zkracující) a `casesensitive="1"` (v případě, že pravidlo přesně dbá na velikost písmen zkratek a textů; bez uvedení tohoto atributu nezáleží na velikosti písmen). Každé pravidlo obsahuje dva párové elementy `<abbrev>` a `<text>` (POZOR! Podle definice XML nezáleží na pořadí elementů.). Elementy `<abbrev>` a `<text>` mohou obsahovat pouze text a první definuje zkratku a druhý definuje text, ze kterého nebo na který se zkratka přepisuje.

Reference:

- `http://en.wikipedia.org/wiki/SMS`
- `http://www.dreamfabric.com/sms/default_alphabet.html`

4.4 CSV: CSV2XML

Vytvořte skript pro konverzi formátu CSV do XML. Každému řádku CSV bude odpovídat jeden dodefinovaný párový element (viz parametr `-l`) a ten bude obsahovat elementy pro jednotlivé sloupce (viz parametr `-h`). Tyto elementy pak již budou obsahovat textovou hodnotu dané buňky z CSV zdroje.

Tento skript bude pracovat s těmito parametry:

- `--help` viz výše
- `--input=filename.ext` zadaný vstupní CSV soubor
- `--output=filename.ext` textový výstupní XML soubor s obsahem převedeným ze vstupního souboru
- `-n` negenerovat XML hlavičku[13] na výstup skriptu (vhodné například v případě kombinování více výsledků)
- `-r=root-element` jméno párového kořenového elementu obalující výsledek. Pokud nebude zadán, tak se výsledky neobalují kořenovým elementem, ač to porušuje validitu XML.
- `-s=separator` nastavení separátoru (jeden znak) sloupců na každém řádku vstupního CSV, kromě tisknutelných znaků jako například středník či čárka podporujte i identifikátor TAB pro tabulátor jako oddělovač; implicitní hodnota je znak čárka (`,`).
- `-h` první řádek CSV souboru slouží jako hlavička a podle něj odvoďte jména elementů XML. Všechny nepovolené znaky nahraďte pomlčkou (vznikne-li i tak invalidní XML element, skončete s chybou). První řádek CSV se potom již neobjevuje ve výsledném XML. V případě, že tento parametr chybí, tak budou elementy pro sloupce generovány jako `colX`, kde X je inkrementální čitač od 1.
- `-l=line-element` jméno elementu, který obaluje zvlášť každý řádek vstupního CSV (není třeba řešit kolize jmen elementů při nevhodné definici `line-element`); implicitní hodnota je `row`.

[13] Tradiční XML hlavička je `<?xml version="1.0" encoding="UTF-8"?>`

- `-i` zajistí vložení atributu `index` s číselnou hodnotou (začíná od 1) do elementu `line-element` (tento parametr se musí kombinovat s `-l`).
- `-e` zotavení z chybného počtu sloupců na neprvním řádku tj. každý chybějící sloupec bude doplněn prázdným polem, přebývající sloupce budou ignorovány. Pokud nebyl zadán tento parametr a vstup obsahuje na některém řádku špatný počet sloupců, tak skript ukončete s chybou.

Verze formátu CSV, ač není standardizován, bude uvažována z RFC 4180 (sekce 2). *Stručně*: Záznamy odděleny koncem řádku (CRLF), poslední řádek neukončen CRLF. Na prvním řádku mohou být volitelně popisky sloupců. Pokud to okolnosti vyžadují, tak se řetězce píší v uvozovkách (např. pokud řetězec obsahuje separátor nebo konec řádku). Uvozovky v ouvozovkovaném řetězci se píší jako dva znaky uvozovek vedle sebe. Bílé znaky mimo řetězce se nevynechávají.

Pro účely skriptu je ještě nutné rozšířit definici neterminálu TEXTDATA z RFC 4180, aby akceptoval i UTF-8 znaky s kódem větším jak 127.

Reference:

- Y. Shafranovich: RFC4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files, 2005. Dostupné na `http://tools.ietf.org/html/rfc4180` [citováno 16. 2. 2010]

4.5 JSN: JSON2XML

Vytvořte skript pro konverzi JSON formátu do XML. Každému prvku z JSON formátu (objekt, pole, dvojice jméno-hodnota) bude odpovídat jeden párový element se jménem podle jména dvojice a obsahem podle hodnoty dvojice. Každé pole bude obaleno párovým elementem `<array>` a každý prvek pole bude obalen párovým elementem `<item>` (kolize jmen elementů se zbytem XML dokumentu neřešte).

JSON hodnoty typu `string` a `number` a JSON literály `true`, `false` a `null` budou transformovány v závislosti na parametrech skriptu na atribut `value` daného elementu s odpovídající hodnotou (stejného tvaru jako v JSON vstupu; nezapomeňte na požadavek ouvozovkování atributů v XML) nebo na textový element v případě hodnoty typu `string` a `number` či na párový element (ve zkráceném zápise) `<true/>`, `<false/>` a `<null/>`.

Tento skript bude pracovat s těmito parametry:

- `--help` viz výše
- `--input=filename.ext` zadaný vstupní JSON soubor
- `--output=filename.ext` textový výstupní XML soubor s obsahem převedeným ze vstupního souboru
- `-n` negenerovat XML hlavičku[14] na výstup skriptu (vhodné například v případě kombinování více výsledků)
- `-r=root-element` jméno párového kořenového elementu obalující výsledek. Pokud nebude zadán, tak se výsledek neobaluje kořenovým elementem, ač to potencionálně porušuje validitu XML.

[14]Tradiční XML hlavička je `<?xml version="1.0" encoding="UTF-8"?>`

- `-s` hodnoty dvojic typu `string` budou transformovány na textové elementy místo atributů.
- `-i` hodnoty dvojic typu `number` budou transformovány na textové elementy místo atributů.
- `-l` hodnoty literálů (`true`, `false`, `null`) budou transformovány na elementy `<true/>`, `<false/>` a `<null/>` místo na atributy.
- `-e` zotavení z chybějícího obalení pole kořenovým objektem tj. globální pole bude obaleno kořenovým objektem (nutno kombinovat s parametrem `-r`; jinak vrátí chybovou hlášku).

Verze formátu JSON, ač není standardizován ale je podmnožinou jazyka JavaScript (Standard ECMA-262, 1999), bude uvažována z webové stránky `http://www.json.org/`. *Stručně*: Soubor obsahuje jeden globální objekt. *Objekty* jsou zapsány ve složených závorkách a obsahují čárkami oddělené dvojice *jméno : hodnota*. *Jméno* je řetězec jazyka JavaScript. *Pole* je zapsáno do hranatých závorek a obsahuje hodnoty oddělené čárkami. *Hodnota* může být další objekt či pole nebo řetězec či číslo jazyka JavaScript. Hodnotou může být také literál `true`, `false` nebo `null`. Prázdné znaky mimo řetězce mohou být ignorovány.

Reference:

- `http://www.json.org/`

5 Závěr

Na stránkách předmětu IPP a v příspěvcích na fóru předmětu IP se budou v prvních týdnech řešení projektu objevovat doporučení a návody na vhodné řešení zadání. Tato doporučení však nebudou závazná a jejich nedodržení je vhodné obhájit v dokumentaci projektu.

Projekt z předmětu IPP si klade za cíl vás seznámit se dvěma skriptovacími jazyky a možnostmi kombinování různých programovacích paradigmat. Dále bychom rádi studenty přesvědčili o vhodnosti různých jazyků pro různé úlohy. Formou diskuzí na fóru potom budeme částečně simulovat komunikaci Vás návrhářů a programátorů se zadavatelem či zákazníkem.

V samotném závěru bych chtěl zdůraznit, že všechny úlohy budou opravovány automatizovaně a tudíž je nezbytné stoprocentní splnění formálních požadavků jak na formáty vstupů a výstupů, tak i na samotné transformace. Pro usnadnění precizního pochopení zadání bude zveřejněna sada příkladů vstupů a výstupů jednotlivých skriptů.