Západočeská univerzita v Plzni
Fakulta aplikovaných věd
katedra informatiky a výpočetní techniky

ZÁPOČTOVÁ PRÁCE Z UIR

Převod mezi čísly v desítkové soustavě, římských číslicích a pomocí vyjádření v českých slovech

20. května 2007

Martin Sloup, A04372
Skupina ÚT 9.20 – 11.10

Martin Sloup, A04372
Skupina ÚT 9.20 – 11.10

Zadání

Vytvořte znalostní systém, který bude umět převádět čísla zapsaná v desítkové soustavě do češtiny a do vyjádření v římských číslicích (a naopak). V případě římských číslic pracujte do čísla 4000. U češtiny a desítkové soustavy pracujte do čísla 1000000000.

Analýza úlohy

V převodu mezi římskými číslicemi a desítkovou soustavou, nebylo potřeba nic složitého vymýšlet. Gramatika vstupního řetězce je jednoduchá. Na rozdíl od toho pro převod z českých slov na hodnotu v desítkové soustavě je gramatika rozsáhlá, číselné reprezentace obsahují skloňování, formát psaní čísel na složenky neobsahuje mezery. Často se můžeme setkat i se specialitami typu „tři a dvacet“ apod. Proto bylo nutné vymyslet komplexní algoritmus, který by si s tím poradil.

Popis algoritmu řešení

Program obsahuje dva algoritmy. První převádí do soustavy římských číslic a druhý provádí převod mezi českými slovy a čísly v desítkové soustavě.

Římské číslice

Pro převod z římských číslic na čísla v desítkové soustavě jsem použil jednoduchý algoritmus, kdy projíždím celý vstupní řetězec. Narazím-li na známý znak (nalezen v tabulce literálů), tak ho převedu na ekvivalentní číslo v desítkové soustavě. Pokud hodnota literálu je menší než hodnota předchozího literálu ve vstupním řetězci, pak odečtu od výsledné hodnoty hodnotu literálu, jinak hodnotu literálu přičtu.
Pokud se ve vstupním řetězci objeví neznámý římský literál, algoritmus vyhodí výjimku a skončí.

Při převodu z čísla v desítkové soustavě na římská číslice provádím nejprve detekci tisíců, kdy zapíšu do výstupního řetězce literál „M“ (tisíc) tolikrát, kolikrát je tisíců ve vstupním řetězci. Následně provedu detekci stovek, desítek a jednotek (v cyklu). Pro příklad uvedu, jak se to bude provádět pro desítky, pro stovky a jednotky je to téměř stejné až na hodnoty literálů. Nejprve tedy zjistím, kolik vstupní řetězec obsahuje desítek. Hodnotu vydělím deseti, abych zjistil příslušný počet desítek. Pokud jich je tam 9, pak provedu zápis „XC“. V případě 4 desítek „XL“. Pokud počet desítek nejsou tato čísla, a je těch desítek víc jak 4, pak zapíšeme hodnotu „L“. Nakonec zapíšeme příslušný počet „X“ (deset), který zbývá (s ohledem na to, že pokud jich je víc jak 5, tak jich zapíšeme o pět míň, z důvodu, že jsme již zapsali těch pět desítek jako literál „L“).

Česká slova

Na převod z českých slov na čísla v desítkové soustavě jsem vymyslel tabulku, která obsahuje 4 skupiny slov. Osobně bych je pojmenoval jako: sčítací (př.: „dva“, „dve“, „pat“, …), náctiné (př.: „nact“), násobné (př.: „set“, „desat“, „sto“, „ste“, „sat“, …) a nakonec násobné opakovací (př.: „tisic“, „milion“, „milarda“). Násobné opakovací z toho důvodu, že se v textu mohou objevit vícekrát a mohou před sebou obsahovat předchozí tři skupiny.
Tedy algoritmus projede vstupní řetězec a hledá, zda neobsahuje tyto skupiny slov. Pokud na nějakou narazí, provede podle příslušné skupiny určitou akci.
U sčítací skupiny si pouze zapamatuje příslušné číslo, které detekované slovo představuje (např.: pro slovo „pet“ je to číslo 5).

U náctiné skupiny přičte k zapamatovanému číslu číslo deset (čímž rozezná výrazy „pat-nact“, kde „pat“ je ze sčítací skupiny, které již detekoval v předchozím kroku).
U skupiny násobné provede vynásobení předchozího zapamatovaného čísla příslušným násobkem (např.: „dve-ste“, kde „dve“ je ze sčítací skupiny, které bylo detekováno v předchozím kroku).
A nakonec pro násobnou opakovací skupinu provede násobení předchozího napočítaného čísla příslušným násobkem v závislosti na tom, jaké číslo bylo detekováno. Výsledek si uloží zvlášť a vynuluje obsah proměnné, která obsahovala předchozí čísla.
Tím je vesměs celý algoritmus hotov, jen ještě obsahuje různé úpravy, tak aby dokázal počítat skupinu násobnou a násobnou opakovací bez zadané hodnoty ze skupiny sčítací. Případně obsahuje ošetření, že když vstupní text obsahuje „pet a pet“, tak jej vyhodnotí jako dvě slova „pet“, které sečte, čímž dokáže rozeznat výrazy „pet a tricet“. To jestli je v textu uvedena spojka „a“, případně mezera nemá na slovech vliv. Taktéž pokud zadaný text obsahuje části slov, případně znaky, které nejsou obsaženy ve zmíněných 4 skupinách, tak ty části (případně znaky) jsou přeskočeny. Tabulka algoritmu dále obsahuje i nespisovné výrazy, takže je možnost detekce například výrazů „sedum“ a „osum“.

U převodu z čísla v desítkové soustavě na slovo se postupuje tak, že je nadefinovaná tabulka slov a jen se zjišťuji příslušná čísla na pozicích v dekadickém čísle a nahrazují se příslušným slovem.
Pokud obsahuje vstupní číslo, číslo větší, než 999, provede se cyklus, kdy se postupně zjistí, kolik obsahuje vstupní číslo tisíců, milionů a miliard (v cyklu). Příslušný počet se tedy pošle rekurzivně znovu do tohoto algoritmu, aby se vrátila příslušná hodnota reprezentovaná slovy. Za tuto hodnotu se přidá příslušné slovo, tj. „tisíc“, „milion“ a „miliarda“ (samozřejmě dáme pozor, aby příslušné slovo bylo správně vyskloňované).
Dále po tomto cyklu a podmínce se pokračuje dále, kdy už se opravdu zjišťuje jen číslo na příslušné pozici a nahrazuje se jeho slovnou reprezentací. Jen je nutné dát pozor na hodnoty od 11 do 19. Tady je nutné provést nahrazení s koncovkou „nact“, tak aby to bylo správně po české stránce.

Programátorská dokumentace

Za programovací jazyk jsem zvolil Javu od SUNu. Měl jsem sice možnost zvolit i programování v C, případně v C++, ale Java je mi trochu více bližší.
Slova a římské literály jsem umístil do pole z důvodu, že se vněm dalo snadno pohybovat. Jinak jsem použil klasické Javovské typy, jako: int, String a boolean. Jen v případě převodu ze slov na dekadické reprezentace jsem použil StringBuffer, protože jsem se potřeboval pohybovat v zadaném vstupním textu a zároveň si to, co jsem již v textu nepotřeboval, odmazávat.
Program jsem nakonec opatřil možností detekce vstupu, tak že není při startu programu udávat, jestli je vstup slova / slovo, římské číslo, či číslo v dekadickém formátu. Program dokáže, pokud není zadaný vstupní text jako argument programu, načítat ze standardního vstupu. Taktéž je možné zadávat čísla ve formě slov a nechat si je opravovat do správného formátu.

Uživatelská dokumentace

Program výsledný program se neinstaluje. Stačí ho jen spustit. Na to je potřeba mít nainstalovaný JRE 1.5 SE od SUN Microsystem.

Program se spouští následovně:

```
java -jar konvertor.jar [zdroj][cíl][atributy] [vstupní text]
```

Jako parametr zdroj a cil, lze použít následující:

- n - převod z čísla / na číslo
- r - převod z římských literálů / na římské literály
- w - převod z českých slov / na česká slova

Jako atributy:

- h, ? - nápověda programu
- b - použití pravidel výpisu na složenky při převodu na česká slova

Pokud nebude uveden parametr zdroj, program se pokusí o autodetekci vstupního textu. Nebude-li uveden vstupní text, program se bude snažit načítat ze standardního vstupu. Program rozeznává pouze česká slova bez diakritiky, to z důvodu, že není možné zaručit správnost diakritiky při vstupu z příkazové řádky, případně jeho výstupu.

Ukázky použití:

Převedení čísla „123456“ na reprezentaci pomocí českých slov:

```
java -jar konvertor.jar nw 123456
```

Převedení římského čísla „IX“ na číslo v dekadickém formátu:

```
java -jar konvertor.jar rn IX
```

Převedení textu „dveste ticet tri“ na číslo (oba způsoby zápisu jsou možné):

```
java -jar konvertor.jar wn dveste tricet tri
java -jar konvertor.jar wn "dveste tricet tri"
```

Načtení vstupního souboru ze standardního vstupu a převod na reprezentaci pomocí českých slov (každá hodnota musí být na zvláštním řádku):

```
java -jar konvertor.jar w < vstup.txt
```

Úryvek ze souboru vstup.txt:

```
12584
MXI
dva a DVACET
jednostotisicdvestepadesatjedna
```

Rozbor výsledků, zhodnocení

Program provádí přesně to, co bylo uvedeno v zadání této semestrální práce. K tomu jsem ještě přidělal možnost použití pravidel výpisu na složenky při převodu na česká slova. Taktéž i možnost detekce vstupní soustavy (čísla, římská čísla a čísla reprezentovaná českými slovy). Dále i možnost vstupu ze standardního vstupu namísto čtení pouze z příkazové řádky. To se hodilo hlavně na testování.

Hlavním problémem programu je, že nedokáže pracovat s diakritikou. V případě načítání vstupu z příkazové řádky, se dá říct, že to není problém, ale v případě načítaní ze standardního vstupu, se může jednat o podstatný nedostatek.

Závěr

Podle zadání jsem vytvořil program, který převádí mezi čísly v desítkové soustavě, římskými čísly a čísly zapsanými pomocí českých slov. U českých slov jsem přidal možnost detekce výrazů jako například „tri a dvacet“, čímž jsem prakticky nechtěně implementoval možnost sčítání („jedna a jedna“, „deset a milion“, …).
V programu, jak už bylo dříve uvedeno, by bylo potřeba vyřešit problémy ohledně vstupního a výstupního kódování při převodu na česká slova, případně z českých slov. Momentálně program totiž rozeznává pouze znaky bez diakritiky.
Program byl úspěšně vzdáleně otestován na operačním systému SunOS s JRE 1.5.0, který se nachází v učebně 412 na katedře informatiky a výpočetní techniky. Taktéž jsem ho otestoval i pod OS Linux Debian 2.6.19.2 (eryx) s verzí JRE 1.5.0_06-b05.