Program csr (Czech SoRt) – abecední řazení podle normy

Petr Olšák

Nejprve jsem si podle dokumentace pana Wagnera k programu `csindex` udělal soukromou verzi programu na abecední řazení. Pak mi kolega Josef Tkadlec půjčil normu ČSN 01 0181 (Abecední řazení). Jakmile jsem ji otevřel, zhrozil jsem se. Když jsem se vzpamatoval ze šoku, pokusil jsem se svůj program co nejvíce přiblížit této normě a hotový produkt věnovat veřejnosti. Následuje dokumentace k programu. Zajímavý je zřejmě odst. 5, kde na mnoha místech doslova cituji normu, upozorňuji na rozpory a uvádím případné návrhy, jak problémy řešit.

Program `csr` používá speciální čtyřprůchodový algoritmus abecedního řazení. První dva průchody probíhají nad jednotlivými slovy a případné další dva průchody nad obsahem celé položky. Podrobněji viz odst. 3. Na rozdíl od české verze programu `makeindex` (`csindex`) program `csr` řadí pouze řádky souboru bez dalšího zpracování pro formátovací účely. Pracuje s externí tabulkou znaků, kterou lze konvertovat do libovolného kódování a tím provádět řazení v libovolném kódování. Tuto tabulku lze také editorem upravit tak, aby řadicí algoritmus lépe vyhovoval představám uživatele. Dále, na rozdíl od programu `csindex`, program řeší řazení podle více slov (např. příjmení a jméno) a řazení podle více položek (heslo, doplněk, podheslo atd.).

Program se volá se třemi parametry:

```
csr input sorttab output
```

kde `input` je název vstupního souboru, `sorttab` je název souboru s tabulkou znaků, podle kterých se má řadit a `output` je název výstupního souboru. Výstupní soubor může být shodný se vstupním.

Výstup se bude od vstupu lišit v pořadí řádků, jinak všechny texty zůstanou beze změny zachovány. V hlavičce vstupního souboru je možné specifikovat, podle kterých sloupců se má provádět řazení (viz odst. 2).

Program `csr` najdete např. v brněnském archívu `ftp.muni.cz` v adresáři `pub/tex/local/cstug/olsak/csr`. Kromě DOSovské (zkompilované) verze `csr.exe` tam je i zdrojový text v jazyce C, který umožňuje snadno instalovat program pro nejrůznější UNIXy. V UNIXu stačí provést toto:

`dos2unix csr.c csr.c` … konverze na UNIXový formát
`cc csr.c -o csr` … vytvoření běhové verze programu

Protože program vyžaduje tři parametry, je užitečné vyrobit script, například s názvem `csort`, obsahující řádek:

```
csr $1 /uplna/specifikace/souboru/sorttab.iso $2
```

Pak se dá program volat z příkazové řádky bez specifikace tabulky řazení, což je určitě uživatelsky přijatelnější. Dále pomocí pracovních souborů v `/tmp` a příkazu `cat` je možné vylepšit script tak, aby se choval jako filtr. Samozřejmě je nutné přesměrovat `stdout` programu do `stderr` nebo do `/dev/null`. Program samotný se jako filtr nechová. Šikovní administrátoři mohou script dále rozšířit o možnosti přepínání mezi různými kódovacími tabulkami podle přepínačů napsaných v příkazovém řádku.

1. Pevné a volné řádky vstupního souboru

Ve vstupním souboru rozlišujeme řádky pevné a volné. Pevné řádky zůstanou na stejném místě i ve výstupní souboru, zatímco volné řádky podléhají abecednímu řazení. Zhruba řečeno, pevné řádky mohou být na začátku a konci souboru, zatímco „uprostřed“ jsou řádky volné.

Pevné řádky jsou všechny komentované řádky na začátku souboru. Komentovaný řádek je řádek, který má v první pozici komentářový znak, například procento (%) (viz odst. 4). Tyto řádky se bez změny pořadí přepíší do výstupního souboru. Počet takových řádků může být libovolný, tj. i nulový.

První nekomentovaný řádek souboru zahajuje pole volných řádků. Pole volných řádků končí buď koncem souboru nebo komentovaným řádkem. Od tohoto komentovaného řádku až do konce souboru jsou všechny řádky (i nekomentované) pevné.

Poznamenejme, že komentářový znak má speciální funkci pouze v první pozici řádku. Kdekoli jinde se interpretuje jako obyčejný znak, tj. jako každý jiný a může například podléhat i abecednímu řazení.

2. Parametry položek

Pokud v poli pevných řádků (na začátku nebo na konci souboru) bezprostředně za komentářovým znakem následuje slovo `csr`, program přečte z tohoto řádku parametry položek. Například:

```
%csr: 30-50, 10-29, 51-61
```

znamená, že se bude nejprve řadit podle podřetězce (položky) v každém řádku od 30. do 50. sloupce, (například tam je v každém řádku napsáno příjmení), pak podle položky vymezené 10. až 29. sloupcem (třeba tam je křestní jméno) a nakonec podle položky vymezené sloupci 51 až 61 (je tam třeba rodné číslo).

Symboly `-` (minus) a `,` (čárka) jsou ve formátu parametrů položek nutné. Za posledním parametrem se nesmí napsat znak `,` (čárka). Počet položek je maximálně 10.

Nenapíše-li se symbol `-` (minus), pak je daná položka ohraničena koncem prvního slova (tj. prvním výskytem mezery, tabelátoru nebo konce řádku). Například, víme-li, že všechna příjmení v naší databázi se skládají jen z jednoho slova, pak stejný požadavek jako v předchozím příkladu můžeme napsat takto:

```
%csr: 30, 10-29, 51
```

Zde pouze v poli křestních jmen předpokládáme výskyt více slov a proto jsme specifikovali i konec položky pro křestní jména.

Pozor na tabelátory! Program interpretuje tabelátor jako mezeru, tj. jediný znak. Program nemůže mít představu, jaká interpretace tabelátoru je v daném případě zvolena. Doporučuje se proto nejprve nahradit všechny tabelátory příslušným počtem mezer a pak teprve určovat parametry položek. Jinak to nebude řadit správně!

V parametrech položek může být bezprostředně za znakem `-` místo číslice znak `$`. Tím je řečeno, že položka končí až koncem řádku. Například

```
%csr: 1-$
```

znamená, že každý řádek od začátku do konce bude brán jako jediná položka. Toto nastavení je implicitní a použije se tehdy, pokud chybí v poli pevných řádků specifikace parametrů položek.

3. Algoritmus řazení

3.1. Algoritmus porovnává jednotlivé řádky nejprve podle položek. Položka je souvislý podřetězec řádku, který se obvykle specifikuje pomocí zadání počátečního a koncového sloupce (viz odst. 2). Položek může být více (maximálně deset). Pokud se shledá obsah první položky z hlediska řazení shodný, přejde se na porovnání druhé položky, pak třetí atd.

3.2. Algoritmus porovnává jednotlivé položky postupně po jednotlivých slovech (zleva doprava). Slovo je skupina znaků, která je od dalšího slova oddělena mezerami a/nebo tabelátory. Mezer a tabelátorů může být mezi slovy libovolné množství (počet mezer a tabelátorů mezi slovy na řazení nemá vliv). Pokud se např. první slovo shledá z hlediska řazení stejné, přejde se na porovnávání druhého slova v rámci položky atd.

3.3. Algoritmus porovnává jednotlivá slova nejvýše ve dvou průchodech. V každém průchodu prochází slovo postupně zleva doprava. V prvním průchodu rozlišuje pouze podle primární řadicí platnosti (tj. nerozlišuje např. ý od y ani od Y. Přesně je to specifikováno v tabulce `sorttab`). Pokud jsou z tohoto pohledu slova stejná, provede se druhý průchod, kde se rozlišují i znaky se sekundární řadicí platností (např. rozlišuje mezi ý a y, ale nerozlišuje mezi y a Y). V tabulce `sorttab` je (podle normy) uvedeno, že v prvním průchodu se nerozlišují písmena s diakritikou (až na čtyři výjimky), zatímco v druhém se diakritika rozlišuje. Ani v jednom průchodu se nerozlišuje mezi velkými a malými písmeny.

3.4. Pokud se všechna slova v jedné položce jeví podle odst. 3.3 jako stejná, provedou se ještě nejvýše dva další průchody, ovšem globálně nad celým obsahem položky, nikoli po jednotlivých slovech. Průchody probíhají zleva doprava a ignorují mezery a tabelátory mezi slovy. V prvním (celkově třetím) průchodu se rozlišují všechny znaky uvedené v tabulce `sorttab` s výjimkou tzv. skoro-prázdných znaků. Prakticky to znamená, že se rozlišují malá a velká písmena. V posledním průchodu se navíc rozlišují skoro-prázdné znaky, což prakticky znamená rozlišování interpunkčních znamének.

3.5. Mezera (nebo tabelátor) jako první znak za slovem je řazená abecedně dříve, než jakýkoli jiný znak. V souboru `sorttab` nelze pozici mezery ani tabelátoru specifikovat. Např. slovo gram bude abecedně dříve, než slovo gramáž. Jinak řečeno: jsou-li slova shodná až na to, že jedno je kratší, pak to kratší slovo bude řazeno dříve.

3.6. Pojem slovo (viz odst. 3.2) je zaveden pro libovolnou skupinu znaků s výjimkou mezery a tabelátoru. Pokud slovo obsahuje jenom prázdné znaky (tj. znaky neuvedené v řadicí tabulce `sorttab` a pro průchod 1 až 3 též skoro-prázdné znaky), interpretuje se jako tzv. prázdné slovo, které je abecedně dříve než jakékoli neprázdné slovo. Mezi různými prázdnými slovy řadicí algoritmus nerozlišuje (u skoro-prázdných znaků se rozlišuje ve čtvrtém průchodu). Příklad je uveden v odst. 5.11.

Příklady správného řazení slov (symbol „$\prec$“ označuje abecední pořadí):

plagiát	$\prec$ plachta	(protože g $\prec$ ch v prvním průchodu)
pláně	$\prec$ plánička	(protože ě $=$ e $\prec$ i, v prvním průchodu)
plánička	$\prec$ Plánička	(protože p $\prec$ P ve třetím průchodu)
Plánička	$\prec$ plaňka	(protože i $\prec$ k v prvním průchodu)
plaňka	$\prec$ plankton	(protože a $\prec$ t v prvním průchodu)
platno	$\prec$ plátno	(protože a $\prec$ á v druhém průchodu)
plátno	$\prec$ platnost	(protože platno $\prec$ platnost v 1. průchodu)
plášť	$\prec$ plat	(protože š $\prec$ t v prvním průchodu)
plat	$\prec$ plát	(protože a $\prec$ á v druhém průchodu)

Celkové řazení:

plagiát $\prec$ plachta $\prec$ pláně $\prec$ plánička $\prec$ Plánička $\prec$ plaňka $\prec$ plankton $\prec$ plášť $\prec$ plat $\prec$ plát $\prec$ platno $\prec$ plátno $\prec$ platnost

Jiný příklad, potvrzující skutečnost, že velká a malá písmena se rozlišují nikoli po jednotlivých slovech, ale v celé položce:

a $\prec$ abc $\prec$ ABC $\prec$ abc frézaře $\prec$ abc nástrojáře $\prec$ ABC nástrojáře

Vysvětlení:
ABC $\prec$ abc frézaře, protože druhé slovo je v položce „ABC“ prázdné a je tedy abecedně dříve, než slovo „frézaře“. K rozlišení dojde už v prvním průchodu nad jednotlivými slovy, kdy se ještě nerozlišují velká a malá písmena.

4. Tabulka sorttab

V této tabulce jsou uvedeny všechny znaky, podle kterých se provádí řazení (včetně speciální notace pro dvojhlásku CH). Do řádků jsou seskupeny znaky, které se nerozlišují v prvním průchodu, do skupin (oddělených mezerou) jsou zapsány znaky, které se nerozlišují v druhém průchodu a všechny znaky se rozlišují v průchodu třetím (s výjimkou skoro-prázdných znaků). Skoro-prázdné znaky jsou uvedeny na zvláštních řádcích uvozených mezerou. Ty se rozlišují ve čtvrtém průchodu. V tabulce `sorttab` se také specifikuje komentářový znak.

Znaky, které nejsou vůbec uvedeny v tabulce `sorttab` jsou tzv. prázdné znaky. Řadicí algoritmus je ignoruje.

Podrobnější informaci o formátu tabulky `sorttab` najde čtenář v dokumentaci k programu (`csr.doc`). Domnívám se, že tyto technické záležitosti nepatří na stránky tohoto časopisu.

Dodávaná tabulka `sorttab` má následující vlastnosti.

- Dvojhlásky ch, Ch a CH jsou mezi h a i.
- V prvním průchodu se rozlišují jen písmena neakcentovaná a písmena č, ř, š a ž.
- V prvním průchodu se též rozlišují číslice 0 až 9, které jsou uvedeny až za písmenem ž.
- V druhém průchodu se rozlišují všechna akcentovaná písmena, ale nerozlišuje se mezi velkými a malými písmeny.
- Ve třetím průchodu se rozlišuje mezi malými a velkými písmeny. Velké písmeno je později, než malé.
- Skoro-prázdné znaky (čtvrtý průchod) jsou uvedeny až za číslicemi a to v tomto pořadí: `.,;?!:"‘’|/()[]<>@&%=+*`

5. Odchylnosti chování programu od normy ČSN 01 0181 a návrhy na řešení

5.1. Norma člení tzv. prvky řazení nikoli na položky, ale na tzv. heslo (např. Novák Jan), doplněk hesla (např. senior), první podheslo (např. název díla), druhé podheslo (např. rok vydání) apod. Heslo, doplněk apod. se může skládat z více slov. Pro různé struktury dat se používá různé členění. Při použití programu můžeme provést např. toto ztotožnění:

heslo = 1. položka
doplněk = 2. položka
1. podheslo = 3. položka
2. podheslo = 4. položka
atd.

Ve výjimečných případech, kdy doplněk hesla má speciální způsob řazení nebo speciální grafickou úpravu, nastávají problémy.

5.2. V článku 7. normy je velmi nejasná formulace týkající se rozlišování slov, která se po prvním průchodu jeví jako stejná. Citujeme doslova (v hranatých závorkách je vysvětlení kontextu, které jsem doplnil):

„*Při jinak zcela stejném slově* [tj. po prvním průchodu] *se řadí nejprve písmena bez diakritických znamének a pak písmena s diakritickými znaménky neobsažená v abecedě podle čl. 5* [čl. 5 uvádí písmena bez diakritiky plus písmena č, ř, š, ž].

Při diakritických znaménkách na různých místech v jinak zcela totožných slovech se řadí nejprve slovo s diakritickým znaménkem blíže konci slova. Vyskytnou-li se nad týmž písmenem slova různá diakritická znaménka, stanoví se pořadí podle čl. 8 [čl. 8 stanoví např. é $\prec$ ě, nebo ú $\prec$ ů $\prec$ ü]. *Vyskytnou-li se slova s diakritickým znaménkem nad týmž písmenem, avšak jedno slovo má ještě další diakritické znaménko, řadí se nejprve slovo s jedním diakritickým znaménkem.*“

Je vidět, že věda se dá dělat ze všeho. Méně průstřelnou formulaci by člověk těžko vymýšlel. Jedna interpretace této části normy velmi blízko odpovídá algoritmu implementovanému do programu (prochází se i v druhém průchodu zleva doprava). Například:

sténá	$\prec$	stěna	(é $\prec$ ě, podle algoritmu v programu `csr`)
sténá	$\prec$	stěna	(nejprve slovo s diak. znaménkem blíže konci slova)
sténá	$\prec$	stěna	(dvě různá diak. znaménka – čl. 8)
sténá	$\succ$	stěna	(nejprve slovo s jedním diak. znaménkem)
nadívá	$\prec$	nádiva	(a $\prec$ á, podle algoritmu v programu `csr`)
nadívá	$\prec$	nádiva	(nejprve slovo s diak. znaménkem blíže konci slova)
lalálá	$\prec$	lálalá	(a $\prec$ á, podle algoritmu v programu `csr`)
lalálá	?	lálalá	(není podle normy rozhodnutelné !?!)

Důsledek: Přesné požadavky normy nejsou do programu implementovány, protože tyto požadavky jsou

a) nejasně formulovány a vedou ke sporným případům,
b) vedou k nerozhodnutelným případům,
c) vyhodnocování zleva doprava je vesměs každému člověku nejbližší.

5.3. Třetí průchod se podle normy dělá nikoli nad jednotlivými slovy, ale nad celými hesly. Tato vlastnost je do programu implementována. Algoritmus seřazování je ale opět trochu zvláštní. Cituji normu, čl. 9:

„*Velká a malá písmena mají stejnou řadicí platnost. Teprve tehdy, liší-li se v psaní velkých a malých písmen dvě HESLA (popřípadě další prvky řazení), shodná jak v písmenech standardizované abecedy, tak v diakritických znaménkách, řadí se nejprve heslo s malým písmenem. Různí-li se hesla velkým písmenem na různých místech, řadí se nejprve heslo s velkým písmenem blíže konci hesla. Při různém počtu velkých písmen se řadí nejprve heslo s menším počtem velkých písmen.*“

Můžeme učinit tyto důsledky:

aBCD $\prec$ AbcD (a $\prec$ A podle algoritmu v programu `csr`)

aBCD $\succ$ AbcD (tři velká písmena jsou více než dvě podle normy)
AbcD ? aBcD (není podle normy rozhodnutelné !?!)

Požadavek normy na srovnávání podle počtu velkých písmen není do programu implementován z podobných důvodů, jako v odst. 5.2.

5.4. V dokumentu [2] upozorňuje pan Wagner na potřebu čtvrtého průchodu pro rozlišení dvou variant velké alternativy písmene CH (tj. Ch a CH). Já jsem to v normě nenašel a považuji za postačující, rozlišit Ch $\prec$ CH už ve třetím průchodu.

Pan Wagner se podrobně rozepisuje o problému „když Ch není Ch“. Jeho příklad se slovem „mochnátý“ je jeden z nejpěknějších příkladů. Při použití programu `csr` je nutno vložit mezi c a h v takových případech prázdný znak, tj. např. `moc-hnátý` nebo `moc{}hnátý`. Tím se potlačí interpretace c a h jako dvojhlásky ch. Přitom řadicí algoritmus prázdné znaky ignoruje. Nebo jiný příklad. `C.~H.~CH.` bude při prázdných znacích `~` a `.` (vlnka a tečka) řazeno jako `CHCH`, ale první dvojice CH bude interpretována jako Cé a Há a druhá dvojice jako CH (Cvičná Horská Chata).

5.5. Německé ostré ß se řadí jako `ss`.

Tato vlastnost (a mnoho podobných) se dá řešit zařazením preprocesoru a postprocesoru. Nechť např. znak `#` není uveden v tabulce `sorttab`, tj. jedná se o prázdný znak. Nahraďme preprocesorem výskyty ostrého ß skupinou tří znaků `ss#`. Pak provedeme řazení, které skutečně řadí `ss#` jako `ss`, protože `#` je prázdný znak. Nakonec postprocesorem nahradíme skupinu znaků `ss#` ostrým `s`. Takové pre- a post-procesory se např. v UNIXu implementují jednoduše jako příkaz `sed` s určitými parametry.

V TEXu nemáme vůbec žádný problém, protože tam se ostré ß zapisuje jako `\ss{}`, tj. stačí mít znaky `\`, `{` a `}` jako prázdné.

Je třeba si uvědomit, že preprocesor prodlužuje texty položek, tj. může se stát, že položky už nebudou správně pod sebou zarovnané ve sloupcích. Pokud se jedná jen o pár pozic (mezer), většinou se nic neděje, protože program přeskakuje libovolné množství mezer na začátku položky.

5.6. Některé znaky (např. řecké γ) se řadí jako slovo (tj. „gamma“) a nikoli jako znak.

V takovém případě můžeme použít preprocesor podobně jako pro ostré ß. V TEXu opět nemáme problémy, protože tam se např. znak γ píše jako `$\gamma$`. Stačí mít `$` a `\` jako prázdné znaky.

5.7. Problém s číslicemi a čísly komplikuje norma takto:

„Číslice se řadí podle svého významu v hesle nebo v dalších prvcích řazení:

a) buď jako slovo (resp. sousloví), pokud jsou jako část delšího slovního výrazu chápány spíše jako slovní součást oficiálního nebo ustáleného označení určitého jevu, události, názvu apod.

b) anebo v číselném pořádku, jestliže výrazně označují pořadí něčeho, chronologii apod.

c) jestliže je výjimečně nutno přihlížet přednostně k číselné hodnotě číslic, řadí se ve vzestupném číselném pořadí až za abecedu, tj. za písmeno ž.“

Příklady:

- zz $\prec$ z-2 $\prec$ ž $\prec$ 3 (podle c) – je implementováno v tabulce `sorttab`.
- 1 $\prec$ 10 $\prec$ 100 (podle b) – platí 001 $\prec$ 010 $\prec$ 100. Stačí nahradit vedoucí nuly před číslem nějakým znakem, který v dalším zpracování nebudeme tisknout. Např. v řádku s nulou v tabulce `sorttab` stojí znak `_`. Pak využijeme toho, že: `__1` $\prec$ `_10` $\prec$ `100`.
- 10 pohádek $\prec$ 2. místo $\prec$ 25 let $\prec$ 1. výročí (podle a) – řadí se jako: deset pohádek $\prec$ druhé místo $\prec$ dvacet pět let $\prec$ první výročí.
- Václav III $\prec$ Václav IV (protože Václav 3 $\prec$ Václav 4).

Poslední dva příklady nelze snadno implementovat. Většinou jsou dosti výjimečné a pokud je potřeba je řešit, pak jsou dvě možnosti:

1. Preprocesorem nahradíme všechny výskyty číslovek slovy (např. místo 10 bude `\deset`). Programovat takový preprocesor pro obecné použití by asi bylo dost složité.

2. Vytvoříme „stínovou položku“ s obsahem např. „deset pohádek“, podle které se řadí, ale která se nebude tisknout při dalším typografickém zpracování. Máme tedy vedle sebe položky „deset pohádek“ (podle které řadíme) a „10 pohádek“ (kterou tiskneme).

5.8. K interpunkčním znaménkům a značkám (tj.: `!`, `;` apod.) se obecně při řazení nepřihlíží. Teprve v případě, že jsou hesla stejná a liší se jen interpunkcí, pak se přistupuje k řazení podle interpunkce. Podle charakteru řazeného materiálu je možné některá interpunkční znaménka zařadit do tabulky `sorttab` například jako skoro-prázdné znaky. Pro orientaci citujme čl. 14 normy:

„Jestliže se znaménka, značky, obrazce vyskytují samy o sobě ve funkci hesla, řadíme je až na konec abecedy, resp. až za číslice (pokud jsou řazeny podle čl. 11 c) [viz bod c) v odst. 5.7]*, a to v tomto pořadí:*

`. , ; ? ! :` *všechny tvary uvozovek*

`- | / \ ( ) [ ] < > { }`

& £ § % ‰ $

$= + \times * \# \sim \approx \simeq$

další grafické značky

Stejného pořadí se používá při podřazování smíšených hesel se stejnou písemnou částí a se znaménkem (značkou, obrazcem) na stejném místě hesla.

Ve speciálních případech je nutno určit pořadí podle zásad příslušné vědní disciplíny.“

Poznamenejme, že v dodávané tabulce `sorttab` nejsou uvedeny všechny tyto znaky. Pozměňte si tabulku `sorttab` tak, aby to nejlépe odpovídalo řazenému materiálu.

5.9. Existují skupiny slov, které se mají interpretovat jako jedno slovo. Např. mluvnické členy v románských jménech (la le) se berou dohromady se slovem:

lebeda $\prec$ le Bon $\prec$ lepra $\prec$ l' Herbier $\prec$ Li Pen $\prec$ liška

V těchto případech je možno slova spojovat např. nějakým prázdným znakem, třeba:

lebeda $\prec$ le~Bon $\prec$ lepra $\prec$ l'~Herbier $\prec$ Li~Pen $\prec$ liška

Postprocesor může tyto vlnky nahradit mezerami. V TEXu nemáme problémy.

5.10. U firemních názvů skládajících se ze dvou nebo více slov spojených symbolem `&` nebo `+` nebo `a` se nejprve řadí podle jednotlivých slov bez přihlédnutí ke spojce. Až v případě, že jsou obě (všechna) slova stejná, pak se ke spojce přihlíží.

Zeman + J. $\prec$ Zeman a Zeman $\prec$ Zeman & Zeman $\prec$ Zeman + Zeman $\prec$ Zeman & Ž.

Jak to implementovat? Spojku `a` nahradit např. znakem `@`. Deklarujeme-li v tabulce `sorttab` znaky `@`, `&` a `+` jako skoro-prázdné (v uvedeném pořadí), pak skutečně máme:

Zeman + J. $\prec$ Zeman @ Zeman $\prec$ Zeman & Zeman $\prec$ Zeman + Zeman $\prec$ Zeman & Ž.

Postprocesor pak nahradí znak `@` písmenem `a`.

5.11. Porovnávání prázdných slov (viz odst. 3.7). Příklad:

1. Novák
2. Novák – junior
3. Novák – senior
4. Novák Jan
5. Novák Jan – senior
6. Novák Karel

Pro pochopení řazení těchto údajů je vhodné interpretovat např. první řádek jako řádek s prvním slovem neprázdným (Novák) a s dalšími slovy prázdnými. Druhý řádek má první slovo neprázdné (Novák) a druhé prázdné (–), protože se skládá z prázdných znaků. Třetí slovo je neprázdné (junior). Nerovnost 1. $\prec$ 2. platí, protože prázdné třetí slovo prvního řádku $\prec$ neprázdné třetí slovo druhého řádku. Podobně např. 3. $\prec$ 4., protože prázdné druhé slovo třetího řádku $\prec$ neprázdné druhé slovo čtvrtého řádku. Výsledný efekt je přesně to, co od řazení očekáváme.

Máme-li různě oddělené doplňky hesel, případně z významového hlediska jsou stejná slova jinak interpretovaná, pak je potřeba být ve střehu. Citujme čl. 19 normy (včetně příkladu):

„*Jestliže se sejde v téže abecední sestavě ve funkci pořadatelů* [prvních slov hlavního hesla] *rodinné jméno se stejně psaným osobním jménem, řadí se nejprve osobní jméno.*"

Příklad: Adam - bakalář $\prec$ Adam de la Halle $\prec$ Adam, Alfred.

Program správně seřadí, pokud prostřední prvek píšeme třeba ve tvaru `Adam ~ de la Halle`, tj. vnutíme za osobní jméno `Adam` prázdné slovo. V první položce už prázdné slovo prezentované znakem `-` máme.

5.12. Ještě větší problémy jsou s tzv. dvojitými jmény. Norma praví, že nejprve řadíme rodinná jména jednoduchá a potom teprve dvojitá. Dvojité rodinné jméno se nepovažuje za jedno slovo. Má platit:

Nová, Zdena $\prec$ Nová-Vlachová, Anna $\prec$ Nováček

Problém je možno řešit například tím, že pro rodinná jména (příjmení) sestavíme jinou položku, než pro jména křestní. Takto je to ukázáno v příkladu v odst. 2. Dále pak musíme v rodinných jménech nahradit před zpracováním spojovník mezerou a po zpracování tam vrátit spojovník. To lze dělat automaticky pre- a post-procesorem. Jiným řešením je vložit do tabulky `sorttab` znak `-` dopředu před všechny abecední znaky. To ale předpokládá, že znak `-` nebude v řazeném souboru použit jiným způsobem. Pokud ale znak `-` je použit jako „klasická“ interpunkce, můžeme třeba nahradit všechny spojovníky dvojitých jmen znakem `^`, který bude uveden v čele tabulky `sorttab`. Postprocesor pak nahradí tento znak zpětně spojovníkem.

5.13. K mluvnickému členu na začátku hesla se nepřihlíží. Například „Der grosse Einbruch“ se řadí jako „grosse Einbruch“.

U německo-českého slovníku stačí psát členy do sloupce (položky), ke které se při řazení nepřihlíží, ale která se nakonec tiskne. Taková položka může být „vysunuta vlevo“ od hlavní položky. U samostatného německého slovníku je třeba postupovat podle příslušné normy DIN, kterou neznám. U již zpracovaných databází (například pro názvy děl v knižním katalogu) by asi bylo potřeba vytvořit preprocesor.

5.14. Nechtěl jsem nikoho odradit od entuziasmu algoritmizovat řazení podle české normy. Nicméně je vidět, že problematika je široká a k různým typům datových vzorků je potřeba přistupovat individuálně. Norma byla schválena v roce 1977, tj. před 17 lety. K výpočetní technice je tam napsáno tolik:

„*Norma je určena pro konvenční (ruční) řazení. Doporučuje se však, aby i při počítačovém řazení, zejména při sestavování programů a úpravě vstupních dat bylo k této normě přihlíženo (v mezích daných úrovní strojové techniky) za účelem snížení stupně diskompatibility mezi ručně a strojově řazenými abecedními sestavami na minimální, nevyhnutelnou míru.*“

Není vyloučeno, že v době, kdy píšu tyto řádky, existuje i norma pro strojové řazení. V každém případě se mezi strojovým a ručním řazením musíme snažit „snížit stupeň diskompatibility na minimální, nevyhnutelnou míru“. Ověřte si, jak ctí tuto skutečnost různé, často draze koupené, databázové systémy, které o sobě hlučně tvrdí, že obsahují národní „lokalizaci“. Mám zkušenosti, že mnohdy slavná lokalizace končí prachbídným přeložením manuálu do češtiny a počeštěním nabídek.

6. Omezení programu csr

6.1. Program dynamicky alokuje paměť pro obsah celého vstupního souboru. Tím je omezena velikost zpracování vstupního souboru. Například předkompilovaná verze programu (`csr.exe`) dodávaná v balíku umí pracovat jen s konvenční pamětí, tj. maximální rozměry souborů jsou cca 400 kB. Na druhé straně program byl testován na UNIXových pracovních stanicích (SUN) a bez problémů zpracoval i soubory velikosti několika desítek MB obsahující např. 250 000 řádků. Nepovažuji proto omezení dané velikostí alokovatelné paměti za příliš rozhodující. Určitě též bude možné kompilovat program pro DOS tak, aby pracoval s rozšířenou pamětí. Já to dělat nebudu, protože

a) pracuji s kompilátorem C pro DOS verze 1.0,
b) mám k dispozici zmíněné pracovní stanice.

6.2. Maximální délka jednoho řádku je stanovena parametrem MAXLINE ve zdrojovém textu programu na 1000 znaků. V případě potřeby je možné uvedenou hodnotu zvětšit. Na nároky dynamické alokace paměti to nemá vliv.

6.3. Počet řazených řádků vstupního souboru je omezen prostorem dynamicky alokovatelné paměti a číslem 2 147 483 648.

6.4. V tabulce `sorttab` ani ve vstupním souboru nelze pracovat se znakem ASCII 0, protože tento znak má v programu speciální význam. Také nelze pracovat se znaky ASCII 1 až 4, protože těmto kódům je přiřazena vnitřní reprezentace dvojhlásek ch, Ch, cH a CH. Se všemi ostatními znaky (ASCII 5 až 255) lze pracovat bez omezení. S výjimkou mezery, tabelátoru a znaku/znaků pro konec řádku v daném systému lze tedy pro libovolné jiné znaky v rozsahu 5 až 255 definovat jejich interpretaci v tabulce `sorttab` a pak podle nich provádět řazení.

7. Literatura

[1] ČSN 01 0181 … Norma na abecední řazení slov.

[2] csindex.dvi … český/slovenský index, dokumentace. Český překlad a implementace češtiny: Zdeněk Wagner. Dokumentace je součástí volně šířeného balíku CSINDEX, který je např. přibalen do volně šířeného balíku CSTEX.