VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

FONETICKÁ TRANSKRIPCE ČESKÉHO JAZYKA

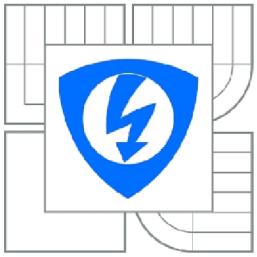
PHONETIC TRANSCRIPTION OF CZECH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ PAVEL
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR SYSEL, Ph.D.
SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Lukáš Pavel ***ID:*** 125280
Ročník: 3 ***Akademický rok:*** 2011/2012

NÁZEV TÉMATU:

Fonetická transkripce českého jazyka

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte používané metody zápisu zvukové podoby jazyka pomocí zvolených sad znaků. Zaměřte se především na specifika českého jazyka a pravidla jeho přepisu. Vytvořte v prostředí Matlab nebo v jazyce C program pro automatický přepis psaného textu do fonetického zápisu a naopak. Funkci programu ověřte na několika testovacích promluvách.

DOPORUČENÁ LITERATURA:

[1] PUSTKA, J.; MÜLLER, L.; MATOUŠEK, J.; RADOVÁ, V. Mluvíme s počítačem mluvenou řečí. 1. vydání. Praha: Academia, 2006. ISBN 80-200-1309-1.
[2] DELLER, J., R.; HANSEN, J., H., L.; PROAKIS, J. G. Discrete-Time Processing of Speech Signals. 2. edition. New York: IEEE Press, 2000. ISBN 0-7803-5386-2.
[3] POKORNÁ, J.; VRÁNOVÁ, M. Přehled české výslovnosti. 1. vydání. Praha: Portál, 2007. 928 s. ISBN 978-80-7367-169-3

Termín zadání: 6.2.2012 ***Termín odevzdání:*** 31.5.2012

Vedoucí práce: Ing. Petr Sysel, Ph.D.
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ANOTACE
Cílem práce je vytvoření programu pro automatický přepis českého jazyka do fonetické podoby. První kapitola této práce se zabývá popisem základních fonetických jednotek, abeced a symbolů, které fonetické abecedy používají. Druhá kapitola podrobně popisuje pravidla fonetického přepisu českého jazyka, kde jsou uvedeny i konkrétní příklady. Nakonec ve třetí kapitole je popis vytvořeného programu v programovacím jazyce C, včetně vývojových diagramů pro lepší orientaci v jeho funkcích.

Klíčová slova: fonetika, transkripce, programování C, český jazyk

ABSTRACT
The goal is to create an application for automatic transcription of Czech language into a phonetic form. The first chapter of this thesis describes the basic phonetic units, alphabets and symbols, that phonetic alphabet uses. The second chapter describes in detail the rules of phonetic transcription of Czech language, including specific examples. Finally, in the third chapter is a description of the application created in C programming language, including flowcharts for better orientation in its functions.

Keywords: phonetic transcription, programming C, czech language

PAVEL, L. *Fonetická transkripce českého jazyka.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 46 s. Vedoucí bakalářské práce Ing. Petr Sysel, Ph.D..

Prohlášení

Prohlašuji, že svoji bakalářskou práci na téma Fonetická transkripce češtiny jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....
podpis autora

Poděkování
Děkuji vedoucímu práce Ing. Petru Syslovi, Ph.D. za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce.

V Brně dne ....
podpis autora

Obsah

Úvod

Fonetická transkripce se zabývá přepisem mluvené formy jazyka do grafické podoby. Výslovnost slov se zapisuje s pomocí speciálních abeced a sad znaků. V této prácí se budu zabývat strojovým zpracováním textu v českém jazyce, pomocí programovacího jazyka C, a jeho překladem do fonetického zápisu.

První kapitola této práce se zabývá popisem základních fonetických jednotek, abeced a symbolů, které fonetické abecedy používají. Druhá kapitola podrobně popisuje pravidla fonetického přepisu českého jazyka, kde jsou uvedeny i konkrétní příklady. Nakonec ve třetí kapitole je popis vytvořeného programu s vývojovými diagramy pro lepší orientaci v jeho funkcích.

1 Základní řečové jednotky a fonetické abecedy

1.1 Základní řečové jednotky

Řeč je sled zvuků, které dohromady vytváří mluvené slovo. Skládá se ze základních jednotek, kterým říkáme hlásky nebo fonémy. Foném a hláska jsou odlišné pojmy. Hláska vyjadřuje souhrn foneticky podobných zvuků, které můžeme vyslovit. Na rozdíl od fonému hláska vyjadřuje jen jediný zvuk, který se nedá zaměnit s jinou hláskou. Jako základní jednotku ji používá Fonetika. Fonetika se zabývá způsobem tvoření zvuků člověkem pomocí artikulačního ústrojí. Ve fonetice nezkoumáme význam vytvořených slov, a proto se hlásky zkoumají napříč všemi jazyky stejně. Fonetický zápis zapisujeme do hranatých závorek [ ]. Jeden foném muže být vyjadřován více hláskami, které mohou změnit význam slova, třeba hlásky [a] a [á] v češtině jsou dva fonémy /a/ a /á/, které udělají rozdíl ve významu mezi slovy *závěs* a *zavěs* (třeba telefon) nebo [s] a [š] ve slovech *sedák* a *šedák*, ale například hlásky [m] a [ɱ] jsou v češtině jeden foném /m/ protože, když zaměníme jeden za druhý tak se nezmění význam slova, například tramvaj. Foném jako základní jednotku používá fonologie. Fonologie se na rozdíl od fonetiky zabývá významem slov, a proto se rozlišuje mezi různými jazyky. Fonologický zápis zapisujeme mezi lomítka / /.[1]

1.2 Fonetické abecedy

Fonetické abecedy nebo také fonetické inventáře se používají pro zápis mluveného slova pomocí speciálních symbolů vyjadřujících výslovnostní formu. Tento přepis pak nazýváme fonetická transkripce.

1.2.1 Mezinárodní fonetická abeceda IPA

Jak už název napovídá abeceda IPA (International Phonetic Alphabet) je mezinárodní abeceda pro fonetický zápis. IPA obsahuje symboly pro všechny možné hlásky, a proto umožňuje porovnávat výslovnosti mezi světovými jazyky. Tato abeceda používá mnoho doplňujících symbolu pro bližší určení vydávaného zvuku, například rozlišuje délku trvání hlásky na velmi krátké [˘], krátké [ ], středně dlouhé [ˑ] a dlouhé [ː], dále obsahuje symboly pro výšku hlasu, úrovně jsou značeny těmito symboly, velmi vysoká [˥], vysoká [˦], střední [˧], nízká [˨], velmi nízká [˩] také zavádí symboly zvýšení [↑] nebo snížení [↓] výšky hlasu a řadu dalších, v rámci této práce bude využívána česká fonetická abeceda (ČFA) nebo SAMPA a proto nebudu dále rozebírat nepřeberné množství symbolů, které mezinárodní fonetická abeceda nabízí.

1.2.2 SAMPA

Protože se symboly IPA velice špatně pracovalo na počítači, vznikla roku 1989 SAMPA (Speech Assessment Methods Phonetic Alphabet) jako náhrada mezinárodni fonetické abecedy pro zpracování počítačem. SAMPA zakóduje IPA symbol do 7bitů ASCII kódu, které jsou snadně tisknutelné tiskárnou. SAMPA byla nejprve použita pro fonetický přepis šesti evropských jazyků (angličtinu, němčinu, dánštinu, holandštinu, francouzštinu a italštinu). Ze základu kódováni těchto šesti jazyků, se odvozuje kódování pro další jazyky. Mezi základní doporučení patří, aby se symboly malých písmen latinky, které se shodují s IPA, zůstali stejné i v SAMPA a ostatní aby se kódovaly ASCII znaky 33-126. Doporučení

SAMPA se ale nemusí respektovat, například v českém jazyce se kvůli zpřehlednění krátké souhlásky píší jako /a/, /e/, /i/, /o/, /u/, ale podle doporučení SAMPA by se měli zapisovat krátké takto /a/, /E/, /I/, /O/, /U/. Bohužel těmito národními úpravami SAMPA ztrácí výhodu v možnosti porovnání mezi různými jazyky. V roce 2006 byla SAMPA oficiálně formulována pro téměř polovinu Evropských jazyků (včetně češtiny) a několik dalších světových jazyku jako je třeba arabština, turečtina nebo hebrejština. V roce 1995 byla na University of London vytvořena rozšířená varianta abecedy SAMPA a byla nazvána X-SAMPA nebo také extended SAMPA, jejímž účelem je umožnit kódování veškerých symbolů IPA na snadno tisknutelné znaky ASCII. Tímto X-SAMPA umožnila zápis pro všechny možné lidské jazyky. K zápisu nejpoužívanějších znaků se používá jednoho znaku a jsou stejné pro obě abecedy (SAMPA, X-SAMPA), k zápisu méně používaných znaku se používá zápis se zpětným lomítkem. Diakritika se zapisuje pomocí podtržítka, za kterým následuje určení typu diakritiky.

1.2.3 Ostatní fonetické abecedy

Samozřejmě existuje i plno dalších abeced, které jsou často vytvořeny jen za účelem fonetické reprezentace konkrétního jazyka tak, aby co nejjednodušeji vyjádřil danou výslovnost. Fonetické symboly se pak mohou zapisovat buď jedním, nebo více znaky. V případě zápisu symbolu s více znaky je nutné oddělovat jednotlivé fonetické symboly mezerou.

1.2.4 České fonetické abecedy

K fonetickému přepisu Českého jazyka se převážně používá Česká fonetická abeceda nebo její zjednodušená veze, ovšem je možné udělat fonetický přepis češtiny pomocí IPA, ale protože IPA není moc vhodná pro přepis slovanských jazyku [2], není u nás moc využívána. Naopak pomocí abecedy SAMPA je možné fonetický přepis českého jazyka velmi dobře vyjádřit a je vhodný pro strojové zpracování. V tabulce 1.1 je zobrazen rozdíl mezi abecedami ČFA, ZČFA, IPA a SAMPA. Česká fonetická abeceda ve své základní formě hojně využívá více znaků pro vyjádření jednotlivých hlásek, což znemožňuje její efektivní čtení, proto se zavedla a často se používá zjednodušená ČFA která pro zpřehlednění používá českou diakritiku, díky tomu se fonetický přepis v obyčejných větách velice podobá samotné větě a je lehce čitelný. Pro prvky, které nelze vyjádřit pomocí znaků z české abecedy, se nejčastěji používají znaky převzaté z abecedy IPA, z čehož vyplývá, že je problém s touto abecedou pracovat v počítači.

Tab. 1.1: Srovnání fonetických abeced v češtině. Tabulka je převzata z publikace [1].

	ZČFA	IPA	SAMPA	ČFA	Příklad		ZČFA	IPA	SAMPA	ČFA	Příklad
vokály	i	ɪ	i	i	*lis*	plozivy	p	p	p	p	*pec*
	e	ɛ	e	e	*pes*		b	b	b	b	*bratr*
	a	a	a	a	*sad*		t	t	t	t	*tuk*
	o	ɔ	o	o	*kov*		d	d	d	d	*dům*
	u	ʊ	u	u	*sukně*		ť	c	c	tj	*děti*
	í	ɪː	iː	ii	*víno*		ď	ɟ	J\	dj	*dítě*
	é	ɛː	eː	ee	*lék*		k	k	k	k	*kost*
	á	aː	aː	aa	*sál*		g	ɡ	g	g	*tygr*
	ó	oː	oː	oo	*kód*	nazály	m	m	m	m	*muž*
	ú	ʊː	uː	uu	*růže*		n	n	n	n	*víno*
diftongy	o͡u	o͡ʊ	o_u	ow	*bouda*		ň	ɲ	J	nj	*laňka*
	a͡u	a͡ʊ	a_u	aw	*auto*	afrikáty	c	t͡s	t_s	c	*cena*
	e͡u	e͡ʊ	e_u	ew	*eunuch*		č	t͡ʃ	t_S	ch	*oči*
frikativy	f	f	f	f	*fík*		d͡z	d͡z	d_z	dz	*podzim*
	v	v	v	v	*vítr*		d͡ž	d͡ʒ	d_Z	dzh	*džbán*
	s	s	s	s	*sůl*	významné alofony	ŋ	ŋ	N	ng	*tango*
	z	z	z	z	*koza*		ɱ	ɱ	F	mg	*tramvaj*
	š	ʃ	S	sh	*škola*		ɣ	ɣ	G		*abych byl*
	ž	ʒ	Z	zh	*žena*		r̝	r̝	Q\	rsh	*tři*
	c͡h	x	x	x	*chata*		ɾ	ɾ	r=		*krk*
	h	ɦ	h\	h	*hůl*		l̩	l̩	l=		*vlk*
	l	l	l	l	*vlak*		m̩	m̩	m=		*osm*
	r	r	r	r	*rok*		ʔ	ʔ	?		*„ráz“*
	ř	r̝	P\	rzh	*moře*		ə	ə	@		*„šva“*
	j	j	j	j	*jev*						

Varianty fonému

Neboli také významné alofony jsou malé změny výslovnosti fonému, které jsou způsobeny přizpůsobením hlasového ústrojí na vyslovení předchozí nebo následující hlásky. Použití v české transkripci není nutné, protože tyto malé změny neovlivňují smysl vět, jen upravují znělost některých písmen. Používají se převážně pro zpřesnění reprezentace řeči. Dále napíšu přehled významných fonému, které se používají v českém jazyce.

- **Zadopatrové n [ŋ]:** zadopatrové *n* se vyslovuje při spojeních *nk* a *ng* pro příklad uvedu slova *tank* nebo *lingvistika*.

- **Retozubné m [ɱ]:** tato varianta fonému /m/ se vyskytuje ve spojeních *mv* a *mf* třeba ve slovech *tramvaj* a *lymfa*.
- **Znělé ch [ɣ]:** znělé *ch* vzniká asimilací znělosti před znělou párovou souhláskou, přiklad *bych dal*.
- **Neznělé ř [r̝̊]:** tato varianta fonému /ř/ vzniká, je-li před nebo za písmenem neznělá hláska např. *křik* anebo se *ř* nachází před pauzou např. *pepř*.
- **Slabikotvorné r [r̩]:** je jedna z dalších variant fonému /r/, která se objevuje v případě, že se před a za *r* nachází souhláska anebo v případě, že se *r* nachází na konci slova po souhlásce, např. *krk* nebo *katr*.
- **Slabikotvorné l [l̩]:** vzniká stejně jako předcházející slabikotvorné *r*, v případě, že se *l* nachází mezi samohláskami anebo na konci slova po souhlásce, např. *vlk*, *dohodl*.
- **Ráz [ʔ]:** jedná se o hlasivkovou explozivu, užívá se před slovem, které začíná samohláskou např. [ʔabeceda] nebo na morfologickém švu ve fonetickém slově např. [trojʔúhelník].

1.3 České řečové jednotky

V následujícím textu se budu zabývat českými řečovými jednotkami. České řečové jednotky se rozdělují na samohlásky a souhlásky. Dále se zde budu zmiňovat o slabikách

1.3.1 Samohlásky

V Češtině se vyskytuje 5 krátkých variant samohlásek /a/, /e/, /i/, /o/, /u/ a 5 dlouhých variant /á/, /é/, /í/, /ó/, /ú/ tzn. celkově 10 samohlásek (y, ý považujeme jako i, í a ů je stejné jako ú). V českém jazyce se dále vyskytují ještě dvojhlásky a to /a͡u/, /e͡u/ a /o͡u/, první dvě dvojhlásky se vyskytují jen ve slovech přejatých, zato dvojhláska /o͡u/ je ryze česká. Dvojhlásky se vyslovuji zpočátku jako hláska prvního písmena (a, e, o), která se plynule změní na hlásku druhou tedy ve všech případech /u/, které je ale částečně potlačeno. Mezi samohlásky patří i neutrální samohláska (šva) [ə], která se objevuje při hláskování *b* [bə], *c* [cə], *d* [də] a dalších.

1.3.2 Souhlásky

V češtině se vyskytuje 27 souhláskových fonémů: /b/, /c/, /č/, /d/, /ď/, /f/, /g/, /h/, /c͡h/, /j/, /k/, /l/, /m/, /n/, /ň/, /p/, /r/, /ř/, /s/, /š/, /t/, /ť/, /v/, /z/, /ž/, /d͡z/, /d͡ž/ a několik významných fonémů. Přesná výslovnost souhlásek je důležitá pro srozumitelnost celého slova.

1.3.3 Slabiky

Slabiky jsou základní zvukové útvary českého jazyka, z kterých se vytvářejí slova. Slabiky tvoří jeden nebo více fonému a jejich nejobvyklejší délka je o dvou nebo třech fonémech. Jako základní jednotky mají důležitý význam jak ve tvoření slov tak jsou důležité zejména pro určování intonace, časování nebo přízvuku.

2 Fonetická transkripce češtiny

Fonetická transkripce, pomocí speciálních symbolů z fonetických abeced, přesně popisuje zvuky mluvené řeči. Podle toho jak věrný chceme přepis udělat, si vybereme i správnou abecedu, při výběru záleží, k jakému účelu bude přepis sloužit, v případně mezinárodní práce se může použít IPA nebo SAMPA, pro srozumitelnější národní formu se hodí spíše ČFA nebo ZČFA, pro zpracování počítačem se hodí použít abecedu SAMPA nebo ČFA. V případě podrobnějšího popisu zvuků můžeme použít významné alofony. V souvislé řeči se fonémy spojují do slabik a mohou se vzájemně ovlivňovat, toto ovlivnění může zapříčinit vznik nového fonému nebo vynechání fonému stávajícího. Tyto změny jsou důležité pro přesný fonetický přepis. Při fonetické transkripci se v zásadě používá spisovné výslovnosti, pro kterou existuji také určitá pravidla. Existuje více pravidel pro správnou spisovnou výslovnost, zejména záleží na výslovnostním stylu. Existují 3 základní výslovnostní styly:

- **Styl zběžný**
  Tento styl se používá v běžné hovorové řeči (pokud je spisovná). V tomto stylu se setkáme s častým zjednodušováním souhláskových spojení, a celkově rychlejším tempem řeči. Zásluhou svižného dění při sportovních zápasech tohoto stylu využívají sportovní komentátoři.
- **Styl vybraný**
  Vybraný styl má pomalejší tempo a vyznačuje se pečlivou výslovností, nejčastěji se s ním setkáme při slavnostních nebo významných projevech.
- **Styl základní**
  Poslední styl je styl základní, tento styl se používá ve veřejných projevech neutrálního charakteru. Můžeme se s ním setkat na odborných přednáškách, ve škole nebo třeba u televizních nebo rádiových hlasatelů. S tímto stylem budu dále pracovat při praktické realizaci práce.

2.1 Automatická fonetická transkripce

Automatická fonetická transkripce se nejčastěji využívá pro přesnější rozpoznávání řeči nebo naopak tvorbu řeči syntetickým zvukem. Fonetický přepis má určitá pravidla, která nazýváme jednoduše fonetická (fonologická) pravidla. Následující definice a struktura je převzata z [1]:

JESTLIŽE řetězci znaků A bezprostředně přechází řetězec znaků C a je bezprostředně následován řetězcem znaků D,

PAK se A přepíše na řetězec znaků B.

Pro jednoduchost budu v této práci toto pravidlo zapisovače tvaru:

$A \rightarrow B / C \_ D$

Řetězce A, C a D představují posloupnosti písmen v běžném textu, řetězec B obsahuje symboly fonetické abecedy. Pro překlad slov, které mají nezvyklou výslovnost nebo jsou přejatá z cizího jazyka se používá fonetický slovník výjimek.

Posloupnost zpracování textu by měla být asi následující:

- Text můžeme zpracovávat buď zleva doprava, nebo zprava doleva, v druhém případě se lépe řeší vícenásobná asimilace znělosti.
- Než začneme slovo zpracovávat, porovnáme ho s výjimkami z fonetického slovníku výjimek, pokud se slovo shoduje, použijeme výjimku ze slovníku.
- Jestliže na slovo nejde použít žádná výjimka, zkusíme vyhledat pravidla, která by se dala na text použít.
- Pokud se v textu nalezne znak, na který neexistuje žádné pravidlo ani výjimka, jednoduše znak opíšeme, nebo ho přizpůsobíme vybrané fonetické abecedě.

2.2 Pravidla fonetického přepisu

Při určování pravidel budu pro jednoduchost používat ZČFA. V následující tabulce jsou popsány symboly, používané pro popis pravidel, které vychází z publikací [1] a [2].

Tab. 2.1: Pomocné symboly pro zápis fonetické transkripce češtiny. Tabulka je převzata z publikace [1].

Značka	Popis	Příklad
V	samohlásky (vokály) a dvojhlásky	⟨[a], [á], [e], [é], …, [u], [ú], [a͡u], [e͡u], [o͡u]⟩
K	souhlásky (konsonanty)	⟨[b], [c], [č], [d], …, [z], [ž]⟩
ZPK	znělé párové souhlásky	⟨[b], [d], [ď], [g], [v], [z], [ž], [h], [d͡z], [d͡ž], [ř]⟩
NPK	neznělé párové souhlásky	⟨[p], [t], [ť], [k], [f], [s], [š], [c͡h], [c], [č], [r̥]⟩
JK	jedinečné souhlásky	⟨[m], [n], [ň], [l], [r], [j]⟩
´	hlavní (slovní) přízvuk	[´jedu ´domú]
|	hranice mezi slovy	[dům | bil]
#	pauza (terminální předěl)	[# … řek̥ # že # přijde … #]
-	Vnitřní předěl (prefixový nebo mezislovní šev)	[v-lese], [na-jíst] atd.
+	vnitřní předěl (sufixový šev)	[buť+me] apod.
¬	znělostní protějšek	¬[b] = [p], ¬[p] = [b], ¬[d] = [t], ¬[t] = [d], …
NP	neslabičné předložky	⟨*k*, *s*, *v*, *z*⟩
JPZ	jednoslabičné předložky (končící znělou souhláskou)	⟨*bez*, *nad*, *ob*, *od*, *pod*, *před* ⟩
*	libovolný symbol	
ø	prázdný symbol	

2.2.1 Základní pravidla

Zde je sepsán souhrn základních bezkontextových pravidel pro fonetický přepis českého jazyka. Bezkontextová pravidla se používají pro přepis českých znaků bez ohledu na kontext

textu, jedná se o přepis českých znaku, pro které v české fonetické abecedě nenajdeme hlásky se shodným zápisem. Jedná se o [y], [ý], [ů] a [ch]. Základní pravidla vypadají takto:

$$y \rightarrow i \; / \; \_$$
$$ý \rightarrow í \; / \; \_$$
$$ch \rightarrow c\underline{h} / \; \_$$
$$ů \rightarrow ú \; / \; \_$$

Příklad: *pytel* [pitel], *být* [bít], *charakter* [charakter], *bůček* [búček].

Ostatní české znaky se v případě ZČFA přepisují stejně, jako se píší v obyčejné české abecedě. V případě použití jiné abecedy je zapotřebí přepisovat i jednotlivé znaky, které se liší od abecedy obyčejné, např. při použití ČFA se *á* zapíše jako [aa], v případě abecedy SAMPA [a:].

2.2.2 Zápis dvojhlásek

Čeština obsahuje tři dvojhlásky *au*, *ou*, *eu*, jejich přepis je následující.

$$au \rightarrow a\underline{u} \; / \; \_$$
$$ou \rightarrow o\underline{u} \; / \; \_$$
$$eu \rightarrow e\underline{u} \; / \; \_$$

Příklad: *kousek* [kousek], *autor* [autor], *eukalypt* [eukalipt].

2.2.3 Spojení samohlásky a souhlásky

Tato spojení se vyskytují jak v rámci jedné slabiky, tak i mezi dvěma slabikami. V rámci jedné slabiky dochází ke změnám ve spojeních [d], [t], [n] se samohláskou [i].

$$d \rightarrow ď \; / \; \_ \langle \; i,í \; \rangle$$
$$t \rightarrow ť \; / \; \_ \langle \; i,í \; \rangle$$
$$n \rightarrow ň \; / \; \_ \langle \; i,í \; \rangle$$

Příklad: *dílna* [ďílna], *tisk* [ťisk], *nic* [ňic].

Ke stejné změně dochází i v případě spojení *dě*, *tě*, *ně*, *mě*.

$$dě \rightarrow ďe \; / \; \_$$
$$tě \rightarrow ťe \; / \; \_$$
$$ně \rightarrow ňe \; / \; \_$$
$$mě \rightarrow mňe \; / \; \_$$

Příklad: *děda* [ďeda], *tělo* [ťelo], *něco* [ňeco], *rozměr* [rozmňer].

V rámci slabiky také dochází ke změnám ve spojeních [b], [p], [v], [f] s písmenem *ě*, kde se *ě* přepisuje na [je].

$$ě \rightarrow je \; / \langle \; b, p, v, f \; \rangle \; \_$$

Příklad: *běh* [bjeh], *pěvec* [pjevec], *vědro* [vjedro], *žirafě* [žirafje].

2.2.4 Výslovnost souhláskových skupin

Souhláskové skupiny jsou v češtině celkem běžné, vyskytují se v množství situací, uvnitř slabiky (např. *sklo*), mezi slabikami (např. *mož-nost*), mezi slovy (např. *návrat domů*), mezi předložkou a slovem (např. *pod-ložit*). Při vyslovení takového slova dochází k asimilaci (sblížení) vlastností sousedících souhlásek, které umožňuje snadnější vyslovení slova.

Asimilace znělosti

Asimilace znělostí umožňuje našemu artikulačnímu ústrojí snadněji přecházet mezi výslovnostmi dvou sousedních souhlásek. Asimilaci znělosti používáme při rozmluvách automaticky. Vlivem asimilace dochází k vyrovnání typu znělosti celé skupiny souhlásek (např. *stovky* [stofky]). Asimilace se dělí do dvou skupin, asimilace regresivní a asimilace progresivní. Při **asimilaci regresivní** (zpětné) rozhoduje o znělosti souhláskové skupiny poslední souhláska. (např. *sbalit* [zbalit], *podložka* [podloška]). Při **asimilaci progresivní** rozhoduje o znělosti souhláska první (např. *nashledanou* [naschledanou]). Progresivní asimilace je v češtině použita zřídka.

Ve spojení dvou a více souhlásek přejímá celá skupina typ znělosti poslední souhlásky skupiny. Děje se tak uvnitř slov, na hranicích souvisle vyslovovaných slov (tj. bez pauzy) i na vnitřním předělu. Zatímco uvnitř slova mohou asimilaci znělosti vyvolat pouze párové souhlásky, na hranici slov asimilaci iniciuje i jedinečná souhláska, samohláska nebo ráz. Na prefixovém švu spodobu způsobují párové souhlásky a ráz. Ke spodobě znělé párové souhlásky na neznělou dochází i před pauzou (po pauze se v češtině výslovnost souhlásek nemění).[1]

$$\text{ZPK} \rightarrow \neg\text{ZPK} \,/\, \_ \langle\, \text{NPK}, \text{-NPK}, \text{-}\,?, \mid \text{NPK}, \mid \text{JK}, \mid \text{V}, \mid ?, \# \rangle$$
$$\text{NPK} \rightarrow \neg\text{NPK} \,/\, \_ \langle\, \text{ZPK}, \text{-ZPK}, \mid \text{ZPK} \rangle$$

Příklad: *stovka* [stofka], *leckdo* [ledzgdo], *lev* [lef], *hněď* [hňeť], *krb* [krp], *drozd* [drost].

V ostatních případech ke spodobě znělosti nedochází, tj. souhláska si zachovává svou znělost (resp. neznělost) [1].
Příklad: *hřbet* [hřbet], *změna* [zmňena], *sleva* [sleva], *spád řeky* [´spád ´řeky], *smíchat* [smíchat] [1].

Mezi pravidly pro asimilaci znělosti se vyskytují i výjimky:

Mezi tyto výjimky patři výslovnost skupiny souhlásek *sh*, tato skupina se může vyslovovat zněle podle pravidel asimilace, ale i nezněle, pokud se použije postupná asimilace. V Čechách se převážně používá neznělá varianta, na Moravě je častěji používána varianta znělá. Výjimkou jsou slova *shora*, *shůry*, *shluk* a jejich varianty, v kterých je spisovná jen znělá varianta [zhora], [zhůry], [zhluk].

$$\text{sh} \rightarrow \text{zh} \,/\, \_$$
$$\text{sh} \rightarrow \text{zch} \,/\, \_$$

Příklad: *shromáždění* [schromážďeňí] i [zhromážďeňí], *shrnutí* [schrnuťí] i [zhrnuťí].

Mezi další výjimky patří foném /v/, jehož znělostní protějšek je foném /f/. Foném /v/ se v okolí souhlásek chová jako jedinečná souhláska. Nezpůsobuje asimilaci znělosti poslední znělé párové souhlásky a dokonce na hranici slov vyvolává spodobu znělosti poslední znělé párové souhlásky na konci předchozího slova [1].

$$v \rightarrow f \,/\, \_\ NPK$$
$$NPK \rightarrow NPK \,/\, \_\ \langle\ v, -v, \mid v\ \rangle$$
$$ZPK \rightarrow \neg ZPK \,/\, \_\ \mid v$$

Příklad: *vteřina* [fteřina], *kvadrant* [kvadrant], *švestka* [švestka] *hned vedle* [´hnet ´vedle].

Zvláštní postavení mezi českými souhláskami má foném /ř/, který nemá na fonologické úrovní neznělý protějšek. Na fonetické úrovni, ale existuje znělostní pár: znělé [ř] a neznělé [r̝̊]. Tento pár se chová podobně jako ostatní znělostní páry, akorát u něho dochází jak k regresivní spodobě znělosti, tak i ke spodobě progresivní.[1]

$$ř \rightarrow r̝̊ \,/\, NPK\ \_$$

Příklad: *předmět* [pr̝̊edmňet], *třídit* [tr̝̊íďit].

V případě, že končí kmen slova znělou souhláskou párovou a přípona začíná jedinečnou souhláskou (např. *buďme*, *hleďme*, *snažme se* citace[1]), se doporučuje výslovnost s neznělým protějškem na konci kmene, ale je přípustná i výslovnost se znělou souhláskou, která není považována za nespisovnou.

$$ZPK \rightarrow ZPK \,/\, \_\ +JK$$
$$ZPK \rightarrow \neg ZPK \,/\, \_\ +JK$$

Příklad: *buďme* [buďme] i [buťme], *hleďme* [hleďme] i [hleťme], *snažme se* [´snašme se] i [´snažme se] [1].

Jednoslabičné předložky zakončené znělou párovou souhláskou si při vnitřním předělu zachovávají svou znělost, následuje-li nejen znělá párová, ale i jedinečná souhláska [1]. V předložce *přes* se koncovka [s] vyslovuje jako [z] [přez].

$$ZPK_1 \rightarrow ZPK_1 \,/\, JPZ\ \_\ \langle\ -ZPK_2, -JK\ \rangle$$

Příklad: *nad hlavou* [nad hlavo͡u], *pod švestkou* [pot švestko͡u], *přes silnici* [přes silňici], *přes bahno* [přez bahno].

Neslabičné předložky (NP) se obecně před párovou souhláskou na začátku následujícího slova řídí pravidly asimilace znělosti. Výslovnost neslabičných předložek před slovem začínajícím jedinečnou souhláskou [v] už je poněkud složitější [1]. Obzvláště předložka *s*, může být vyslovena dvojím způsobem, a to buď [s] anebo [z]. V případě, že se předložka vyskytuje před osobním zájmenem, je vždy vyslovována jako [s].

$$z \rightarrow z \,/\, \mid\ \_\ \langle\ -JK, -v\ \rangle$$
$$k \rightarrow k \,/\, \mid\ \_\ \langle\ -JK, -v\ \rangle$$
$$v \rightarrow v \,/\, \mid\ \_\ \langle\ -JK, -v\ \rangle$$
$$s \rightarrow s \,/\, \mid\ \_\ \langle\ -JK, -v\ \rangle$$
$$s \rightarrow z \,/\, \mid\ \_\ \langle\ -JK, -v\ \rangle$$

Příklad: *z jablek* [z jablek], *k ledu* [k ledu], *v Praze* [f praze], *s medem* [s medem]i[z medem], *s vámi* [s vámi] nikoliv [z vámi].

Asimilace artikulační

Asimilace artikulační se na rozdíl od asimilace znělosti přizpůsobuje artikulací. Vlivem této asimilace dochází k vyrovnání artikulačních rozdílů mezi sousedními hláskami. Na rozdíl od asimilace znělosti se asimilace artikulační při výslovnosti objevit nemusí. Tato asimilace se nejčastěji objevuje při rychlé mluvě ve zběžném výslovnostním stylu. Dále uvedu soupis pravidel pro přepis:

Hláska [n], která se nachází uvnitř slova před [g] nebo [k], se vyslovuje jako [ŋ]. V tomto případě nemůžeme použít výslovnost s [n], protože je to považováno za nespisovné.

$$n \rightarrow ŋ \,/\, \_ \langle g, k \rangle$$

Příklad: *novinka* [noviŋka], *stránka* [stráŋka], *dabing* [dabiŋg], *anglický* [aŋglický].

Hláska [m], která se nachází uvnitř slova před [f], [v], se může vyslovovat jako [ɱ] nebo [m].

$$m \rightarrow ɱ \,/\, \_ \langle f, v \rangle$$
$$m \rightarrow m \,/\, \_ \langle f, v \rangle$$

Příklad: *tramvaj* [tramvaj] i [traɱvaj], *nymfa* [nimfa] i [niɱfa].

Hlásky [t] a [d], se uvnitř slova před [ň] mohou změnit na [ť] a [ď].

$$t \rightarrow t \,/\, \_ ň$$
$$t \rightarrow ť \,/\, \_ ň$$
$$d \rightarrow d \,/\, \_ ň$$
$$d \rightarrow ď \,/\, \_ ň$$

Přiklad: *závodní* [závodňí] i [závoďňí], *včetně* [včetňe] i [včeťňe].

Podobně jako v předchozím pravidlu lze uvnitř slova před [ť] a [ď] nahradit [n] za [ň].

$$n \rightarrow ň \,/\, \_ \langle ť, ď \rangle$$
$$n \rightarrow n \,/\, \_ \langle ť, ď \rangle$$

Příklad: *pondělí* [poňďelí] i [ponďelí].

Při spojení [t] nebo [d] s hláskou [s] se může vyslovit [c] za předpokladu, že se nezmění význam slova. Spojení [ts] se musí vyslovit, pokud se tyto dvě hlásky nachází na hranici slov nebo na švu předpony nebo předložky. Nesmíme zapomenout na asimilaci znělosti ve spojení [d] a [s], kde dochází ke změně [d] na [t].

$$ts \rightarrow c \,/\, \_$$
$$ts \rightarrow ts \,/\, \_$$
$$t\text{-}s \rightarrow ts \,/\, \_$$
$$t \mid s \rightarrow ts \,/\, \_$$

Příklad: *studentský* [studentskí] i [studenckí], *dětství* [ďetství] i [ďectví], *předsazený* [přetsazení] nikoliv [přecazení], *odstavec* [otstavec] nikoliv [octavec].

Podobně jako v předchozím případě se, při spojení [t] nebo [d] s hláskou [š] může vyslovit hláska [č]. Spisovná výslovnost platí stejně jako u předchozího pravidla, tedy musí se vyslovit, pokud se tyto dvě hlásky nachází na hranici slov nebo na švu předpony nebo předložky. Stejně tak platí i asimilace znělosti [dš] na [tš].

tš → č / _
tš → tš / _
t-š → tš / _
t | š → tš / _

Příklad: *většina* [vjetšina] i [vječina], *mladší* [mlatší] i [mlačí] *dostat šach* [dostat šach] nikoliv [dostatčach].

Při spojení [t] a [d] se [z] nebo [ž] ve většině případů používáme plnou výslovnost, ale je možné v některých případech použít [d͡z] případně [d͡ž].

dz → d͡z / _
dz → dz / _
d-z → dz / _
d | z → dz / _
dž → d͡ž / _
dž → dž / _
d-ž → dž / _
d | ž → dž / _

Příklad: *džungle* [džungle] i [d͡žungle], *odznak* [odznak] i [od͡znak], *podzim* [podzim] i [pod͡zim], *chytit žábu* [chiťit žábu] nikoliv [chiťid͡žábu].

Zjednodušená výslovnost souhláskových skupin

Zde sepíšu pár pravidel pro zjednodušenou výslovnost, pro kterou není definitivní určení, zdali se jedná o výslovnost spisovnou, či nespisovnou.

U přídavných jmen v prvním pádě množného čísla mužského rodu s koncovkou *-ští* , jejichž kmen je zakončen *-z* nebo *-ž* [1], je dovolena i zjednodušená výslovnost.

zští → šťí / _ |
zští → sšťí / _ |
žští → šťí / _ |
žští → sšťí / _ |

Příklad: *francouzští* [francou͡sšťí] i [francou͡sťí], *pražští* [prašš͡ťí] i [praští] [1].

Ve skupině hlásek [šťk], je možno vyslovit i [štk].

ť → t / š _ k
ť → ť / š _ k

Příklad: *tloušťka* [tlou͡šťka] i [tlou͡štka], *klíšťky* [klíšťky] i [klíštky] [1].

U skupin [zdn] a [zdň], je přijatelná i výslovnost bez [d].

$$d \rightarrow d / z \_ \langle n, ň \rangle$$
$$d \rightarrow ø / z \_ \langle n, ň \rangle$$

Příklad: *hvězdný* [hvjezdní] i [hvjezní], *prázdný* [prázdní] i [prázní].

Zjednodušení se dále může provézt ve spojení předpony *vz-* s hláskou [b] nebo [p] pokud nedojde ke změně významu slova.

$$vz \rightarrow z / \langle \mid, - \rangle \_ \langle -b, -p \rangle$$
$$fs \rightarrow s / \langle \mid, - \rangle \_ \langle -b, -p \rangle$$

Přiklad: *vzpoura* [fspo͡ura] i [spo͡ura], *vzbouřit* [vzbo͡uřit] i [zbo͡uřit].

Dále uvedu soupis zjednodušení výslovností konkrétních slov.

Speciální pravidla pro zjednodušování platí pro slova *dcera* a *srdce* a jejich odvozené varianty (např. *dceřiná*). Tyto slova se spisovně vyslovují bez *d* (*t*), tj. [cera] a [srce]. Jejich plná výslovnost se používá, jen vybraném výslovnostním stylu.

Další slova, která se mohou zjednodušovat, jsou tvary slovesa *býti*, kde je možno vypustit hlásku [j] : *šel jsem* [šel jsem] i [šel sem], *kreslili jsme* [kreslili jsme] i [kreslili sme], pokud je sloveso *býti* v záporném tvaru, hláska [j] se vyslovuje (*nejsi* [nejsi]).

V číslovkách *sedm* a *osm* a jejich odvozených variantách je možno vyslovovat i [u]: *sedm* [sedm̥] i [sedum], *osm* [osm̥] i [osum], *sedmdesát* [sedm̥desát] i [sedumdesát].

Zjednodušená výslovnost je přípustná dále ve slovech *ctnost* [ctnost] i [cnost] a *egyptský* [egiptský] i [egipský], kde se vypouští [t].

Slova *přijď* a *přijďte*, se mohou vyslovit i bez [j] tzn. [pr̥ijť] i [pr̥iť], [pr̥ijťte] i [pr̥iťte].

Ve slovech *džber* a *džbán* je možno krom [d͡ž] a [dž] použít i hlásku [ž]. *Džber* [d͡žber], [džber] i [žber], *džbán* [d͡žbán], [džbán] i [žbán].

Výslovnost dvou foneticky stejných souhlásek

Pokud se ve slově vedle sebe vyskytnou dvě foneticky podobné souhlásky (např. *s* a *z*, *b* a *p*), je krom plné výslovnosti možné vyslovit i zjednodušenou variantu.

Uvnitř slova je možné použít jak plnou tak i zjednodušenou výslovnost, obzvláště na švu přípony.

$$** \rightarrow ** / \_$$
$$** \rightarrow * / \_$$

Příklad: *vítězství* [víťesství] i [víťeství], *nižší* [nišší] i [niší], *vyšší* [višší] i [viší].

Pokud není uvnitř slova znatelně cítit jeho skladba, doporučuje se používat zjednodušená výslovnost. V tomto případě by plná výslovnost zněla přehnaně pečlivě.

** → * / _

Příklad: *výkonný* [víkoní], *denní* [deňí].

Ve slovech ve kterých by se zjednodušením výslovnosti změnil význam, se doporučuje požívat plnou výslovnost:

Ve tvarech podstatných a přídavných jmen typu *racci* (racek) – *raci* (rak), *pecce* (pecka) – *pece* (pec), *křečči* (křeček) – *křečí* (křeč) apod. [1].

V příponách rozkazovacího způsobu zakončeného koncovkou *-me*: *oznamme* [oznamme], *uvědomme* [uvjedomme], *zlomme* [zlomme] atd. [1].

Ve spojeních s částicí *-li*: *byl-li* [billi], *znal-li* [znalli], *viděl-li* [viďelli] [1].

Je potřebné používat plnou výslovnost při vnitřním předělu, zjednodušením výslovnosti by zde mohlo dojít ke změně významu.

- → * / _

Příklad: *nejjasnější* [nejjasňejší] nikoliv [nejasňejší], *poddůstojník* [poddústojňík] nikoliv [podústojňík], *bezzubý* [bezzubí] nikoliv [bezubí].

Na hranici slov je vyžadována plná výslovnost.

| → * / _

Příklad: *byl léčen* [bil léčen] nikoliv [byléčen], *máš šerpu* [máš šerpu] nikoliv [mášerpu], *polykač čepelí* [polikač čepelí] nikoli [polikačepelí].

2.2.5 Výslovnost slov přejatých

Slova, která nemají původ, v Českém jazyce nazýváme slova přejatá. Tato slova jsou přejata z jiných jazyků, obohacují český jazyk novými názvy věcí a jevů. Přejatá slova mohou být jak zcela počeštěná, u kterých se změnila výslovnost i pravopis (např. *šunka*), také i zcela přejatá, u kterých se shoduje s původním jazykem jak výslovnost, tak i pravopis slova (např. *software*) Čeština obsahuje mnoho takových slov, proto jejich výslovnost respektive správný přepis nemůžeme ve fonetické transkripci zanedbat. Při počešťování cizích slov se dodržují dvě zásady:

- Přízvuk se přesouvá na první slabiku, kde se vyskytuje u většiny českých slov. Například francouzština má přízvuk na slabice poslední, tento přízvuk je tedy nezbytné přesunout na slabiku první.

- Hlásky, které se nevyskytují v české fonetické abecedě, nahrazujeme nejbližšími foneticky podobnými českým hláskami. Všechna česká pravidla, která se vztahují, na spojování hlásek, se uplatňují i na slovo přejaté, tedy asimilace znělosti i asimilace artikulační.

Pro automatický přepis převzatých slov se ve většině případů používá fonetický slovník výjimek, kde se vedle pravopisného tvaru nachází i tvar fonetický. V případě, že program narazí na slovo z fonetického slovníku výjimek, použije fonetický přepis, který je ve slovníku k danému slovu přiřazený.

Pravidla pro přepis samohlásek ve slovech přejatých

Pokud se v přejatém slově vyskytne spojení [i], [í] nebo [y], [ý] s další samohláskou vkládá se mezi tyto samohlásky souhláska [j].

$$i \rightarrow ij \;/\; \_\ V$$
$$y \rightarrow ij \;/\; \_\ V$$

Příklad: *filozofie* [filoyofije], *gymnázium* [gimnázijum], *folie* [folije], *milion* [milijon], *hyena* [hijena].

V opačném případě, když se [i], [í] nebo [y], [ý] vyskytuje za samohláskou, se místo [i], [í], [y], [ý] vyslovuje souhláska [j]. V některých slovech se může vyslovovat i [ji].

$$i \rightarrow j \;/\; V\ \_$$
$$i \rightarrow ij \;/\; V\ \_$$

Příklad: *detail* [detajl], *toroid* [torojid], *algebraický* [algebrajický], *laik* [lajk].

Ve slovech cizího původu se často vyskytuje spojení *di*, *ti*, *ni*, ve kterých se na rozdíl od českých slov nezměkčuje *d*, ale *i* se vyslovuje jako tvrdé.

$$d \rightarrow d \;/\; \_\ \langle\ i,í\ \rangle$$
$$t \rightarrow t \;/\; \_\ \langle\ i,í\ \rangle$$
$$n \rightarrow n \;/\; \_\ \langle\ i,í\ \rangle$$

Příklad: *dikobraz* [dikobras], *nikl* [nikl], *digitální* [digitálňí], *diktát* [diktát], *titul* [titul],

Pravidla pro přepis souhlásek ve slovech přejatých

Při přepisu souhlásek ve slovech přejatých se dodržuje výše psané pravidlo. Hlásky, které se nevyskytují v české fonetické abecedě, nahrazujeme nejbližšími fonetickými hláskami z české fonetické abecedy. Tedy písmena *w* a *q* vyslovujeme jako [v] a [kv].

$$w \rightarrow v \;/\; \_$$
$$q \rightarrow kv \;/\; \_$$

Příklad: *Watt* [vat], *western* [vestern], *Quido* [kvído].

Při přepisu *x* existují složitější pravidla. Záleží, jestli se *x* vyskytuje osamoceno nebo je součásti nějaké předpony např. *ex-*, *neex-*, *koex-* atd. . V případě, že se *x* vyskytuje

v předponě, přepisujeme ho jako [gz] pokud následuje znělá párová souhláska nebo samohláska, pokud následuje neznělá nebo jedinečná souhláska, přepisujeme *x* jako [ks].

$$ex \rightarrow egz / \langle \mid , - \rangle \_ \langle ZPK, V \rangle$$
$$ex \rightarrow eks / \langle \mid , - \rangle \_ \langle NPK, JK \rangle$$

Příklad: *exekuce* [egzekuce], *expedice* [ekspedice], *existence* [egzistence], *expert* [ekspert].

Jestliže *x* není součástí předpony, platí pro něj následující pravidla:

$$x \rightarrow ks / \_ \langle NKP, \mid NPK, \mid jk, \mid V, \mid ?, \mid \# \rangle$$
$$x \rightarrow ks / \mid \_ V$$
$$x \rightarrow ks \ / V_1 \_V_2$$
$$x \rightarrow gz \ / \_ \mid ZPK$$

Příklad: *oxid* [oksid], *text* [tekst], *textil* [tekstil], *fax došel* [fagz došel].

3 Praktická realizace

Jak už jsem uvedl dříve, realizace je provedená v programovacím jazyce C, programový kód jsem sepisoval v programu Microsoft Visual Studio 2008. Veškeré funkce programu jsou řešeny skrz konzolové okno aplikace. Pro správné zobrazování českých znaku v programu a jejich následné správné zpracování je nutné mít před kompilací nastavenou kódovací sadu na Central European (DOS) - Codepage 852. Dále všechny vstupní soubory pro volbu 2 v hlavním menu programu musí být v kódové sadě 852 a výstupní soubory s diakritikou, které jsou výstupem volby 4 v hlavním menu, je také soubor zapsaný kódovou sadou 852. Na přiloženém CD je program PSPad, který mimo jiné dokáže převádět různé kódové sady mezi sebou, tedy i do kódové sady 852.

3.1 Úvodní část programu

V této podkapitole je seznam použitých příkazů a jejich knihoven, dále je zde seznam proměnných a popis, k čemu jsou v programu používány.

Seznam použitých knihoven a funkcí

```
stdio.h : fopen
          fclose
          fflush
          fprintf
          printf
          scanf
          gets
          getchar

stdlib.h : system
string.h : strcpy
           strcat
           strcmp
           strlen
           strlwr
```

Seznam použitých proměnných a jejich funkce

`cfa` - v této proměnné jsou uloženy všechny v programu používané znaky nebo sady znaků z abecedy ČFA, které jsou seřazeny do skupin: *V* - samohlásky, *ZPK* - znělé párové souhlásky, *NPK* - neznělé párové souhlásky, *JK* - jedinečné souhlásky a *X* - značí funkční znaky, které se nachází na konci proměnné. Znak hvězdičky „ * “ je používán pro zápis nepodporovaných znaků nebo nesmyslné pozice znaku (např. *hě* > *h**) a dále znak středníku „ *;* “, ten je používán pro informování programu o ukončení zadávání textu k překladu. Důležité je také poznamenat, že kvůli přehlednosti při čtení je za každý znak z abecedy ČFA doplněna mezera, což není v tabulce 3.1 zřetelné.

`sampa` - Tato proměnná se shoduje s proměnnou `cfa` s tím rozdílem, že zde najdeme používané znaky z abecedy SAMPA. Obsah proměnné je znázorněn v tabulce 3.1.

`ceska` - tato proměnná má stejnou strukturu jako proměnné `cfa` a `sampa`, s tím rozdílem, že v jednotlivých polích jsou uloženy znaky české abecedy a na konci této proměnné se nachází několik znaků potřebných pro zpětný přepis, které předešlé dvě fonetické abecedy neobsahují (např. ě nebo ů). Obsah proměnné je také znázorněn v tabulce 3.1.

`abeceda` - Do této proměnné se načítá používaná abeceda (ČFA, SAMPA), kterou si v menu zvolí uživatel.

hlasky - Do jednotlivých polí této proměnné se ukládají jednotlivé hlásky detekované nebo přepsané podle základních pravidel při přepisu do fonetické podoby.

asimilace - V této proměnné jsou uložené hlásky slova z předchozího cyklu programu, které se následovně zpracovává podle pravidel asimilace.

text - Do této proměnné se postupně ukládají jednotlivá slova, která jsou zadána uživatelem nebo jsou přečtena ze zvoleného souboru.

vysledek - Proměnná `vysledek` slouží k uložení výsledného slova z proměnné `asimilace` do jediného textového řetězce a následnému výpisu na obrazovku nebo uložení do souboru.

nazevtxt - Do této proměnné se ukládá uživatelsky zvolené jméno pro cílový textový soubor určený pro zápis.

nazevtxtcteni - Do této proměnné se ukládá uživatelsky zvolené jméno pro zdrojový textový soubor určený pro čtení.

menu - Tato proměnná slouží jen pro ukládání čísla zvolené nabídky v menu.

pruchod - Slouží k počítání průchodů cyklem zpracování slov.

zapistxt - Zde je uložena číselná hodnota 1 - zapisovat, 0 - nezapisovat , pro zápis do textového souboru.

ctenitxt - Zde je uložena číselná hodnota 1 - číst, 0 - nečíst , pro čtení z textového souboru.

x - Tato proměnná určuje počet použitých míst (délku slova) v proměnné `hlasky`.

x2 - Tato proměnná určuje počet použitých míst (délku slova) v proměnné `asimilace`.

Lokální proměnné při zpracování asimilace

druhznaku - V této proměnné je uložen druh aktuálně zpracovávaného znaku, tedy jestli se jedná o *V* - samohlásku, *ZPK* - znělou párovou souhlásku, *NPK* - neznělou párovou souhlásku, *JK* - jedinečnou souhlásku, *P* - pauzu nebo *X* - funkční znak.

druhznakuminuly - Zde je uložen druh předešle zpracovávaného znaku. Zkratky jsou stejné jako u proměnné `druhznaku`.

znakslovo - Zde je uložen druh znaku, kterým začíná následující slovo, který se používá při asimilaci, která může nastat mezi dvěma po sobě jdoucími slovy.

pismeno - V této proměnné je uloženo číslo které ukazuje na určitý znak (hlásku) v proměnné `abeceda`.

Lokální proměnné při přepisu z fonetické do normální abecedy

slovo - do pole této proměnné se ukládají jednotlivé hlásky slova načteného ze souboru

slovo2 - v této proměnné jsou uloženy všechny hlásky předchozího načteného slova.

znakted - V této proměnné je uložen druh aktuálně zpracovávaného znaku, tedy jestli se jedná o *V* - samohlásku, *ZPK* - znělou párovou souhlásku, *NPK* - neznělou párovou souhlásku, *JK* - jedinečnou souhlásek, *P* - pauzu nebo *X* - funkční znak.

znaknasled - Zde je uložen druh následujícího znaku, zkratky se shodují s proměnnou `znakted`

pismeno - V této proměnné je uloženo číslo, které ukazuje na určitý znak (hlásku) v proměnné `abeceda`. Určuje se podle něho druh znaku v proměnné `znakted`.

pismeno2 - V této proměnné je uloženo číslo, které ukazuje na určitý znak (hlásku) v proměnné `abeceda`. Určuje se podle něho druh znaku v proměnné `znaknasled`.

fonznak - pomocná proměnná pro zápis znaků přečtených ze souboru s fonetickým přepisem v abecedě ČFA.

z - Tato proměnná určuje počet použitých míst (délku slova) v proměnné `slovo`.

z2 - Tato proměnná určuje počet použitých míst (délku slova) v proměnné `slovo2`.

Tab. 3.1: Obsah proměnných `cfa` a `sampa`.

	V												
	0	1	2	3	4	5	6	7	8	9	10	11	12
cfa	a	e	i	o	u	aa	ee	ii	oo	uu	aw	ew	ow
sampa	a	e	i	o	u	a:	e:	i:	o:	u:	a_u	e_u	o_w
ceska	a	e	i	o	u	á	é	í	ó	ú	au	eu	ou

	ZPK										
	13	14	15	16	17	18	19	20	21	22	23
cfa	b	d	dj	g	v	z	zh	h	dz	dzh	rzh
sampa	b	d	J\	g	v	z	Z	h\	d_z	d_Z	P\
ceska	b	d	ď	g	v	z	ž	h	dz	dž	ř

	NPK										
	24	25	26	27	28	29	30	31	32	33	34
cfa	p	t	tj	k	f	s	sh	x	c	ch	rsh
sampa	p	t	c	k	f	s	S	x	t_s	t_S	Q\
ceska	p	t	ť	k	f	s	š	ch	c	č	ř

	JK								X		P
	35	36	37	38	39	40	41	42	43	44	45
cfa	m	n	nj	l	r	j	mg	ng	*	;	#
sampa	m	n	J	l	r	j	F	N	*	;	#
ceska	m	n	ň	l	r	j	m	n	*	;	,

	Doplňující znaky						
	46	47	48	49	50	51	52
cfa							
sampa							
ceska	ů	x	ě	y	ý	q	w

3.2 Menu a předzpracování textu při přepisu do fonetického zápisu

Nejprve je hned po spuštění programu načten do proměnné `abeceda` obsah proměnné `cfa`, tedy výchozí abeceda určená pro práci s textem, účel a použití této proměnné bude vysvětleno později. Dále už následuje zobrazení hlavního menu, kde je na výběr šest možností, při vybrání první možnosti tedy, výpis překladu na obrazovku se provede zápis hodnoty 0 do proměnné `zapistxt` a zápis hodnoty 0 do proměnné `ctenitxt`. Výběr druhé možnosti, výpis na obrazovku i do souboru, vyzve uživatele k zadání požadovaného jména souboru včetně požadované přípony. V případě, že soubor existuje, se pokračuje se zápisem za poslední zapsané slovo, pokud soubor se zadaným jménem neexistuje tak se vytvoří. Dále se zde zapíše do proměnné `zapistxt` hodnota 1 a také zápis hodnoty 0 do proměnné `ctenitxt`. Samotný zápis do souboru je realizován až na konci programu.

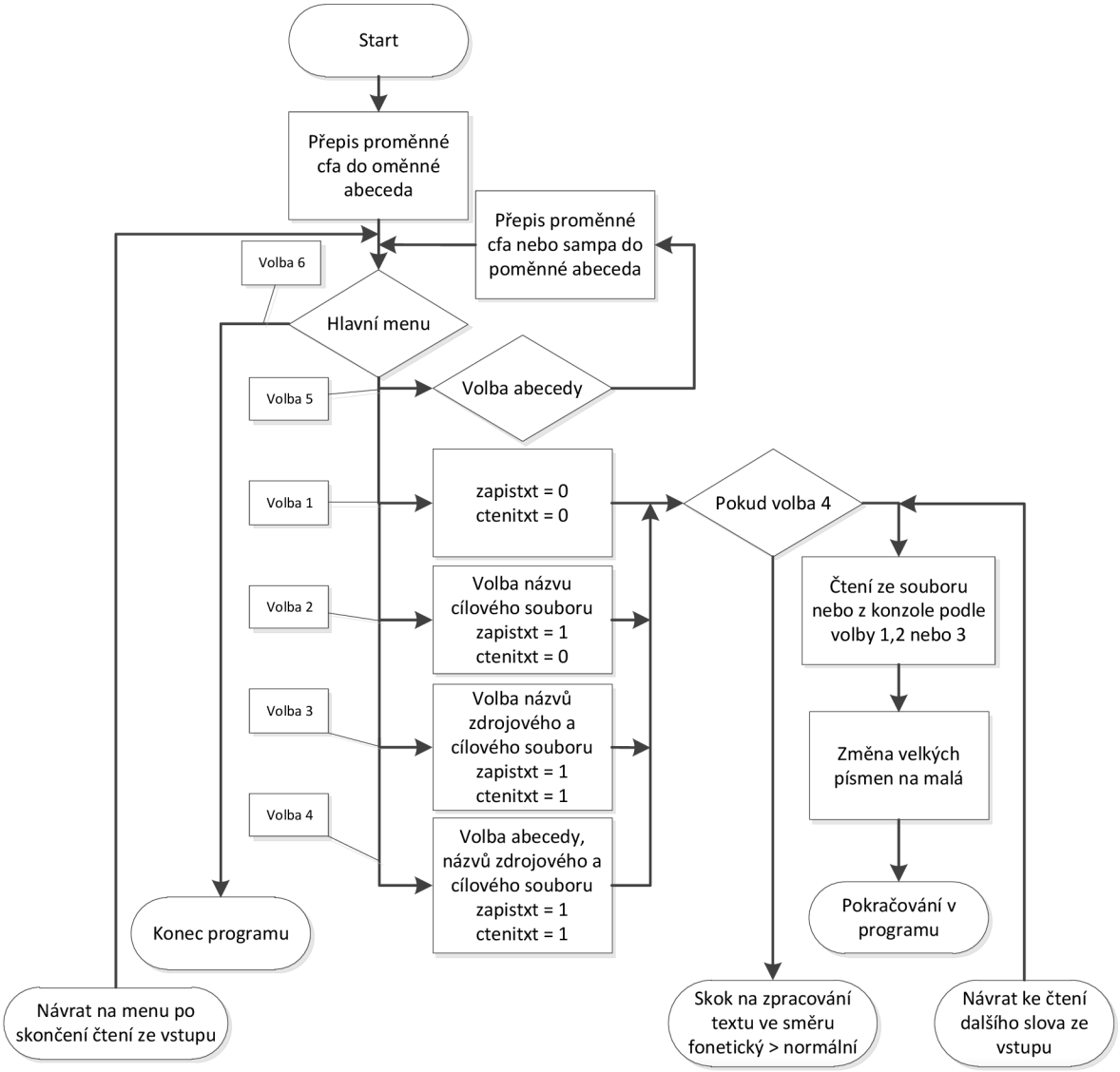
Třetí možností je přímý přepis textu ze zvoleného zdrojového souboru (zdrojový soubor musí být uložen v kódové sadě 852) do zvoleného cílového souboru bez zobrazení přeloženého textu na obrazovku. To se může hodit například pro překlad dlouhých textů. Pokud soubor pro čtení neexistuje, je uživatel informován chybovým hlášením a může se pokusit zapsat jméno souboru pro čtení znovu. Následně je ještě zapsána hodnota 1 do proměnných `zapistxt` a `ctenitxt`.

-Prozatím přeskočím volbu 4 a popíšu volbu 5. Ve volbě 5 je možné zvolit abecedu, se kterou chce uživatel pracovat, možnost výběru je mezi abecedami ČFA a SAMPA. Volbou jedné z těchto položek se cyklem `for` přepíše obsah proměnné `cfa` případně `sampa` do proměnné `abeceda`, se kterou se následně pracuje v programu. Po výběru abecedy je uživatel automaticky navracen do hlavního menu.

-Teď se mohu vrátit k volbě 4, která slouží pro přepis fonetického zápisu zpět do původního textu. Uživatel nejdříve zadá jméno souboru, ze kterého chce číst. Následně je vyzván pro výběr abecedy, ve které je přeložen zdrojový soubor. Výběr abecedy je zde umístěn hlavně z důvodu, aby uživatel nemusel přemýšlet před zvolením volby 4, v jaké abecedě je zdrojový soubor, který chce překládat. Zvolení abecedy probíhá stejným způsobem jako v případě volby abecedy v hlavním menu (volba 5). Následně je ještě vyzván pro zadání jména souboru, do kterého se má výsledný text uložit. Poslední možností (volba 6) je ukončení programu, kde se pomocí příkazu `break` vyskočí z hlavního cyklu. Pokud vybereme jednu z prvních tří možností v hlavním menu, tak se dostaneme do první poloviny programu, kde se zpracovává text směrem z normálního do fonetického zápisu. Do druhé poloviny programu se dostaneme výběrem možnosti 4 v hlavním menu, která způsobí přeskočení první poloviny programu. Druhé polovině programu se budu věnovat později.

Jednotlivé události v běhu programu se u prvních třech možností, mění převážně na základě hodnot v proměnných `ctenitxt` a `zapistxt`. Při volbě možnosti 1 nebo 2 je v proměnné `ctenitxt` uložena hodnota 0, při této hodnotě se na obrazovku zobrazí výzva pro zadání textu. V případě volby 3 je v proměnné `ctenitxt` uložena hodnota 1, výzva se neobjeví a dále se pokračuje v programu. Po této události se dostáváme do hlavního cyklu programu pro zpracování normálního textu do fonetického zápisu.

Následuje jednoduchá podmínka `if`, která rozhoduje podle hodnoty v proměnné `ctenitxt` z jakého zdroje má program číst. Pokud je hodnota 0, program čte z klávesnice příkazem `scanf`, pokud je hodnota 1 program čte příkazem `fscanf` ze souboru zvoleného v hlavním menu. Zároveň se ještě ukládá návratová hodnota příkazu `fscanf` do proměnné `konecsouboru`, která se používá v závěru této části programu pro ukončení čtení a uzavření zdrojového souboru příkazem `fclose` při detekci návratové hodnoty EOF (End of File).



Obr. 3.1: Vývojový diagram pro menu a předzpracování textu.

V případě zadávání textu přes klávesnici je důležité poznamenat, že program kvůli pravidlům asimilace, kdy může následující slovo ovlivnit slovo předchozí, zpracovává vždy předešlé slovo, a tudíž při posledním zadaném slovu čeká na další, aby mohl zkontrolovat, jestli toto další slovo nemůže ovlivnit slovo aktuálně zpracovávané. Proto je nutné nějakým způsobem program upozornit, že uživatel zadal celý text a chce ho přeložit. Toho je dosáhnuto zápisem třech středníků bez mezer „ ;;; “ jako posledního slova. Program následně přeloží zadaný text a zobrazí ho na obrazovku. Uživatel po jeho přečtení stiskne enter a tím se vrátí do hlavního menu.

Vlastní převod textu do fonetického přepisu probíhá po jednotlivých slovech. Po zadání/přečtení textu k překladu z minulého odstavce a následném potvrzení enterem se pomocí příslušného příkazu přečte první slovo a uloží se do proměnné `text`. Toto slovo je následně předzpracováno tím způsobem, že se všechna velká písmena ve slově zamění za malá, protože ČFA používá pro fonetický zápis jen malé znaky a v abecedě SAMPA může velký znak vyjadřovat jinou hlásku (např. z>z | ž>Z). Pro zmenšení klasických ASCII znaků je použit příkaz `strlwr`. Pro zmenšení znaků s českou diakritikou jsem použil vlastní cyklus, který zkontroluje, jestli se ve slově nevyskytuje nějaký z velkých znaků s diakritikou, který se případně přepíše na jeho malou variantu.

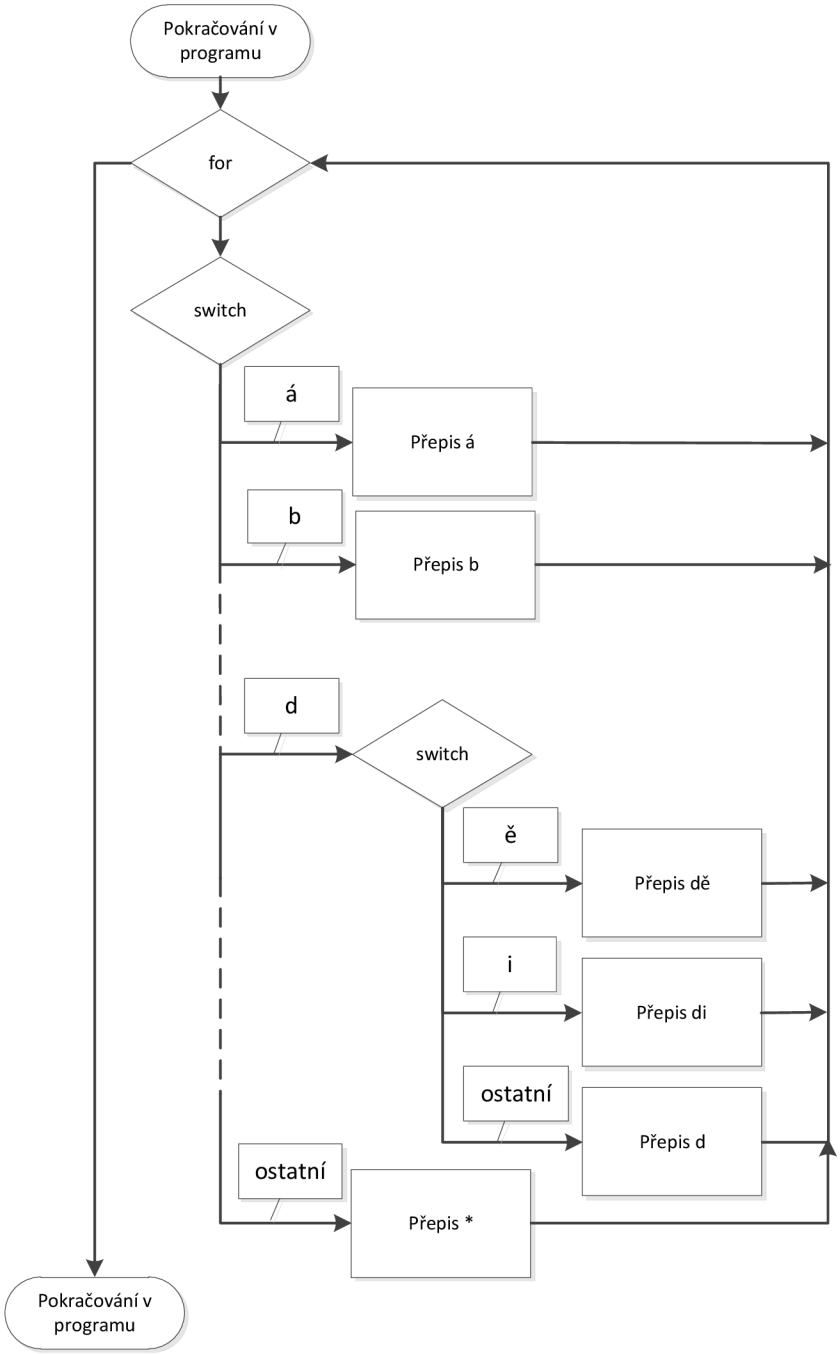
3.3 Aplikace základních pravidel

Po zmenšení písmen se už slovo dostává ke zpracování podle základních fonetických pravidel. Nejdříve se pomocí příkazu `strlen` změří délka slova v proměnné `text`. Tato hodnota se uloží do proměnné `i`, která určuje počet průchodů cyklem `for` a zároveň pomáhá, při určování konkrétního znaku v proměnné `text` viz dále. Následuje přepínač `switch`, který se přepíná podle písmena, které určuje výraz `text[strlen(text)-i]`. Tedy pokud na vstup napíšu například slovo *houba*, proměnná `strlen(text)` bude mít hodnotu 5, tato hodnota bude při prvním cyklu programu stejná s proměnou `i`, do které se hodnota `strlen(text)` uložila a výraz `text[strlen(text)-i]` si můžeme představit jako `text[5-5]`, tudíž výraz ukazuje na první písmeno slova *houba* (`text[0]`). V druhém cyklu se odečte 1 od proměnné `i`, ale hodnota `strlen(text)` je pořád stejná, z čehož vyplývá, že výsledná pozice v proměnné `text` bude `text[1]`, tedy druhé písmeno.

Teď když víme, jak příkaz `switch` určuje, na která písmena se má přepínat, můžeme přejít k samotné aplikaci pravidel. Na obrázku 3.2 je zjednodušený vývojový diagram této části programu. Pro všechny možné znaky české abecedy je v přepínači `switch` příslušný `case`, všechny ostatní znaky jsou nahrazeny hvězdičkou `*`.

Začneme nejjednoduššími pravidly, to jsou taková, kde se vymění znak bez ohledu na to, jaké písmeno se nachází před nebo za ním. Mezi takovéto znaky můžeme pro ukázku zařadit třeba písmena *á* a *b*. Zde už začínáme pracovat s proměnnými `hlasky` a `abeceda`. V případě, že je detekován v proměnné `text` znak *á*, `switch` přeskočí na příslušný `case` pro přepis písmena *á*, kde se provede zápis hodnoty z proměnné `abeceda[5]` (dle tabulky 3.1) do proměnné `hlasky[x]`, kde `x` je pozice v proměnné `hlasky`, která v případě prvního průchodu je 0, tedy první pozice v proměnné. Následně je proměnná `x` inkrementována o 1, čímž se při příštím zápisu zapisuje na následující místo v proměnné `hlasky`. Přepis písmene *b* probíhá obdobně. Dále popíšu, jak se zpracovávají složitější pravidla. Pro příklad uvedu třeba změkčování *d* na *ď* ve spojení s *i*, *í* nebo *ě*. Uvnitř hlavního `case` pro hodnotu *d* je další přepínač `switch`, který kontroluje další písmeno výrazem `text[strlen(text)-i+1]`, kde `+1` je právě určení pozice následujícího písmena, podle kterého se následně program přepne do příslušného `case`. V případě hodnoty *d* je zde `case` pro *i*, *í*, *ě* a hodnota `default` pro všechny ostatní možnosti. Pokud se jako následující hodnota, kterou vyhodnocuje `switch`, objeví například znak *i*, tak se po přepnutí do příslušného `case`

provedou následující operace: do proměnné `hlasky[x]` se na příslušné místo zapíše znak z proměnné `abeceda[15]` (dle tabulky 3.1 je to fonetický ekvivalent písmena ď), inkrementuje se `x`, čímž se posuneme nad následující místo v proměnné `hlasky` a následně do ní zapíšeme obdobným způsobem znak *i* a opět inkrementujeme `x`.



Obr. 3.2: Vývojový diagram pro zpracování základních fonetických pravidel

Protože jsme při této operaci přečetli a přepsali dva následující znaky z proměnné `text`, je nutné se také posunout o jedno místo v hlavním cyklu, proto je zde také dekrementace hodnoty v proměnné `i`, která posune čtení na následující znak v proměnné `text`. Stejným způsobem se přepisují i spojení *dí* a *dě*. Pokud znak následující za *d* není *i*, *í* nebo *ě*, zapíše se do proměnné `hlasky` jen znak *d* stejným způsobem jako jsem výše popisoval zápis znaku *á*.

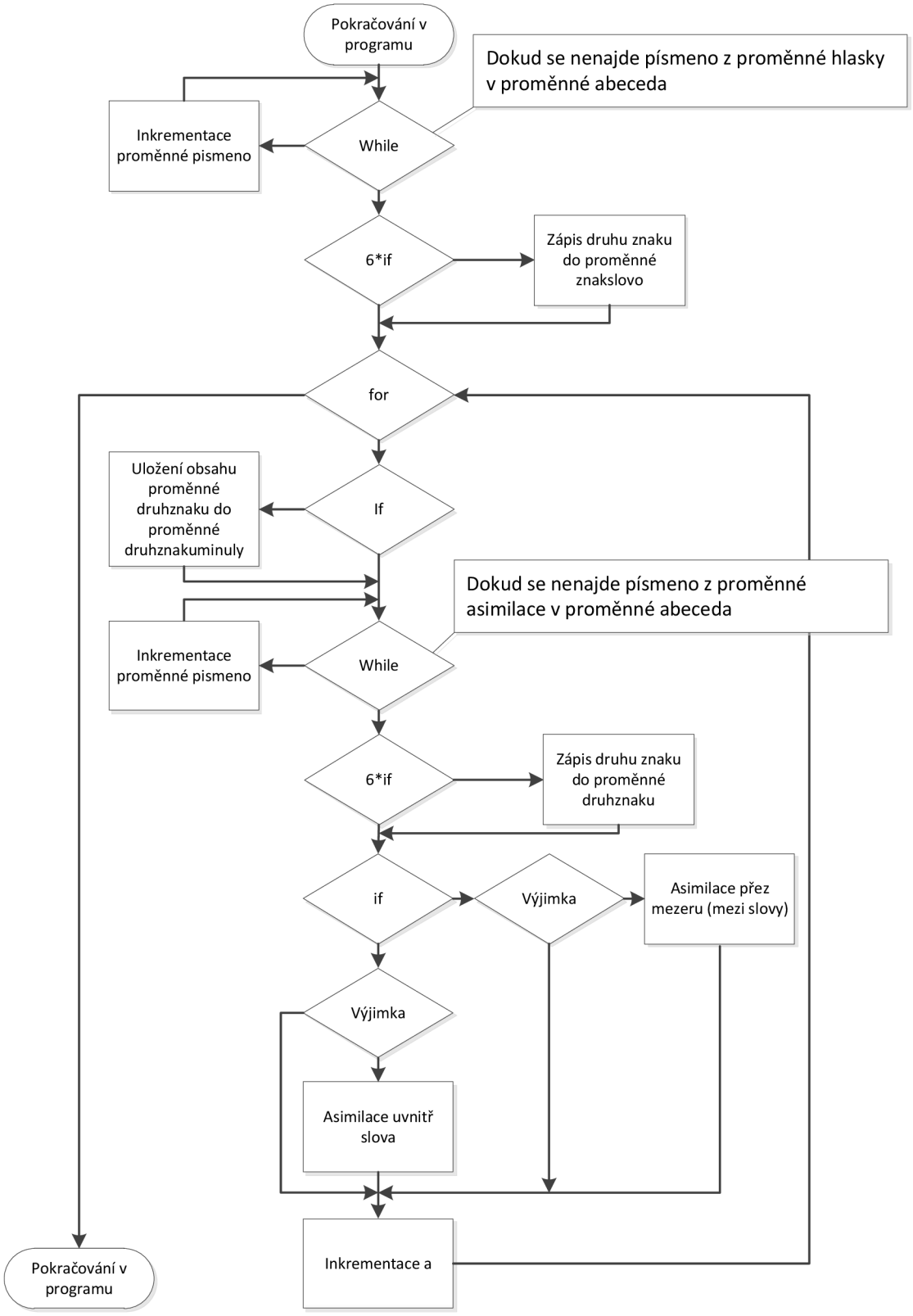
Obdobně jako jsou řešeny spojení *di*, *dí* a *dě*, jsou řešeny i spojení *au*, *ou*, *eu*, *ch*, *ti*, *tí*, *tě*, *ni*, *ní*, *ně*, *mě*. Jediná větší změna je použití `text[strlen(text)-i-1]` u spojení *bě*, *pě*, *vě*, *fě*, kde `-1` kontroluje nikoliv následující, ale na předešlý znak. Předešlý znak se kontroluje, protože při spojeni *bě*, *pě*, *vě*, *fě* nedochází ke změně předešlého písmena jako je například spojeni *dě* (*ďe*), ale změně písmena *ě* na spojení *je*.

3.4 Aplikace pravidel asimilace

Protože při asimilaci může (mimo jiné) následující slovo ovlivnit slovo předešlé, program provádí asimilaci vždy u slova, které bylo zpracováváno v předešlém cyklu. Proto budu v následujícím výkladu předpokládat, že se program již nachází minimálně v druhém cyklu. Zde je důležité poznamenat, že v prvním cyklu došlo k přepsání obsahu proměnné `hlasky` do proměnné `asimilace` současně s uložením délky slova z proměnné `x` do proměnné `x2` (proměnná `x` je následně vynulována). Pokud by tedy program zpracovával např. slovní spojení „*dvě houby*“ tak se v prvním cyklu nachází slovo „*dvě*“ v proměnné `hlasky`, tato proměnná se přepíše do proměnné `asimilace` a na začátku druhého cyklu se do proměnné `hlasky` uloží slovo „*houby*“. To znamená, že na začátku druhého cyklu je první přečtené slovo („*dvě*“) uloženo v proměnné `asimilace` a druhé slovo („*houby*“) je uloženo v proměnné `hlasky`. Na konci prvního cyklu se také inkrementuje hodnota v proměnné `pruchod`, podle které program určuje, zda se má začít s asimilací.

Asimilaci je vhodné zpracovávat od zadu, proto program nejdříve kontroluje, jestli nemůže následující slovo ovlivnit slovo, ve kterém chceme asimilaci provádět. Nejdříve musíme zjistit druh znaku (ZPK, NPK, JK, ...) prvního písmene slova následujícího za zpracovávaným slovem. Na to je použit jednoduchý cyklus `while` s podmínkou:
`strcmp(hlasky[0],abeceda[pismeno])!=0`. Uvnitř cyklu se inkrementuje hodnota v proměnné `pismeno`, která určuje pozici v proměnné `abeceda`. Jakmile je podmínka splněna, program vyskočí z cyklu a v proměnné `pismeno` je uložena pozice písmena v proměnné `abeceda`, tím jsme zjistili, jaké písmeno se nachází v proměnné `hlasky`. Z hodnoty uložené v proměnné `pismeno` můžeme, zásluhou vhodně rozvrženého pořadí hlásek v proměnné `abeceda`, také určit o jaký druh hlásky se jedná. Hodnoty 0-12 jsou samohlásky nebo dvojhlásky (V), hodnoty 13-23 jsou znělé párové souhlásky (ZPK), hodnoty 24-34 jsou neznělé párové souhlásky, hodnoty 35-42 jsou jedinečné souhlásky (JK), hodnoty 43 a 44 jsou funkční znaky (X) a hodnota 45 je pauza (P). Zkratka, která odpovídá zpracovávanému znaku, se zapíše do proměnné `znakslovo`.
Tímto jsme určili druh prvního znaku následujícího slova, tato část programu se nachází před hlavním cyklem zpracování asimilace, do tohoto cyklu vstupujeme až teď. V tomto cyklu nejdříve vynulujeme proměnou `pismeno`, abychom ji mohli znovu využít. Pokud tento cyklus procházíme podruhé, tak se ukládá předchozí hodnota proměnné `druhznaku` do proměnné `druhznakuminuly` (v prvním cyklu používáme místo proměnné `druhznakuminuly`, proměnnou `znakslovo` z následujícího slova). Dále následuje cyklus `while` s podmínkou: `strcmp(asimilace[x2-a-1],abeceda[pismeno]) != 0`, kde v proměnné `asimilace` je uloženo slovo z předešlého cyklu, v kterém určujeme zpracovávané písmeno pomocí výrazu `x2-a-1` (slovo zpracováváme odzadu).

Uvnitř cyklu je opět inkrementace proměnné `pismeno`. Stejným způsobem jako v předešlém případě určíme druh znaku, tedy *V*, *ZPK*, *NPK*, *JK*, *X*, nebo *P*. Jediný rozdíl oproti předešlému případu je, že se příslušná zkratka zapíše do proměnné `druhznaku` nikoliv do proměnné `znakslovo`.



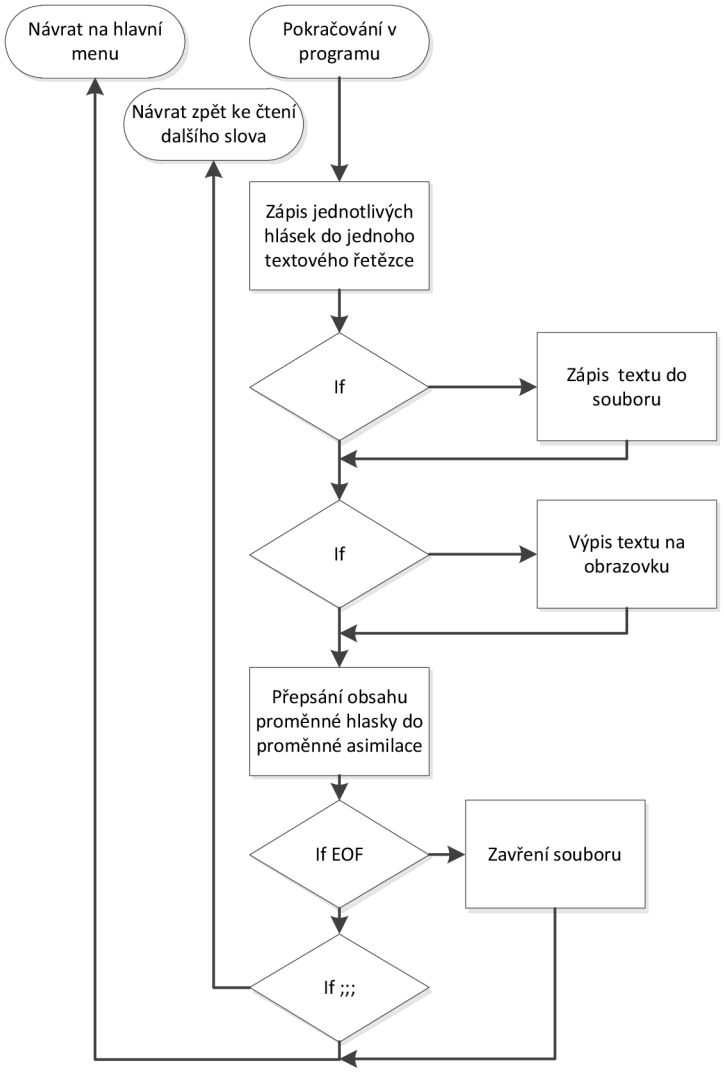
Obr. 3.3: Vývojový diagram pro zpracování asimilace

Když už známe druh zpracovávaného znaku a druh prvního znaku následujícího slova, můžeme začít s asimilací. Při prvním průchodu je použita funkce `if`, v které je zpracovávána asimilace mezi slovy, v každém dalším průchodu program přeskočí do funkce `else`, v které se zpracovává asimilace v rámci slova. V těchto dvou funkcích jsou vnořeny další funkce `if` které vypadají takto:
`if(strcmp(druhznaku,"ZPK")==0 && strcmp(znakslovo,"NPK")==0).`
Následující řádky se provedou, pokud je v proměnné `druhznaku` „ZPK“ a zároveň je v proměnné `znakslovo` „NPK“. Pokud je podmínka splněna zamění se podle fonetických pravidel v proměnné `asimilace` hláska za její znělostní protějšek, toho je dosaženo pomocí výrazu: `asimilace[x2-a-1]=abeceda[pismeno+11],` v proměnné `pismeno` máme uložené aktuální písmeno a jeho znělostní protějšek nalezneme v proměnné `abeceda` (v případě směru ZPK > NPK) přičtením čísla 11, pokud bychom zaměňovali ZPK za NPK, číslo 11 odečítáme. Tím se nám ale změnil druh znaku, proto se do proměnné `druhznaku` zapíše zkratka druhu znaku, na který jsme znak změnili, v našem příkladě je to zkratka NPK. Obdobně jsou řešeny i všechny ostatní pravidla, které jsou sepsána na straně 16 této práce. Při zpracovávání asimilace v rámci slova se neporovnává obsah proměnných `druhznaku` a `znakslovo`, ale `druhznaku` a `druhznakuminuly`. Po zpracování asimilace u aktuálního znaku je na konci cyklu inkrementace hodnoty v proměnné `a`, která určuje, který znak se bude v příštím cyklu zpracovávat. Do určitých pravidel asimilace jsou vnořeny výjimky z těchto pravidel. Například předložka *v* by se podle pravidel asimilace měla změnit na *f*, ale s tou výjimkou, která říká „*v* zůstane *v* pokud následující slovo začíná na *v* nebo jedinečnou souhlásku (JK)“. Ještě musím upozornit, že tato výjimka se aplikuje v pravidle ZPK → ¬ZPK / _ | JK , tedy znělá párová souhláska se změní na její znělostní protějšek, pokud následující slovo začíná na jedinečnou párovou souhlásku. Podobné výjimky jsou řešeny funkcemi `if`. Například funkce pro výjimku v → v / | _ ⟨ -JK, -v ⟩ vypadá takto:
`if(asimilace[0]!=abeceda[17] || x2!=1).` Ve funkci se kontroluje, jestli se první znak proměnné `asimilace` shoduje s písmenem *v*, které se nachází na pozici 17 v proměnné `abeceda`, zároveň se ještě kontroluje délka slova v proměnné `asimilace` (je uložena v proměnné `x2`), aby si byl program jistý, že se jedná o předložku *v*, a ne o slovo které na písmeno *v* začíná. Ke změně znělosti dochází do té doby, dokud se nerovnají zároveň oba výrazy. Jakmile se oba výrazy současně začnou rovnat (slovo začíná na *v* a má délku 1), není splněna podmínka a přeskočí se provedení příkazu pro změnu znělosti. Tím se nezmění *v* ve *f* a je splněna výjimka asimilace.

3.5 Výpis na obrazovku a do souboru

Jakmile jsou ve slově zkontrolovány všechny znaky a případně provedena asimilace podle pravidel, je slovo připraveno na výpis na obrazovku případně do souboru. Nejdříve se obsah proměnné `asimilace`, kde jsou jednotlivé hlásky uloženy samostatně, zapíše pomocí cyklu `for` do proměnné `vysledek`, v které je už slovo uložené jako jeden textový řetězec. Následuje funkce `if` s podmínkou `zapistxt==1` tedy pokud je splněna podmínka provede se zápis obsahu proměnné `vysledek` do souboru, jehož jméno bylo uživatelem definováno na začátku programu. Každé slovo je v cílovém souboru uloženo samostatně na nový řádek.

Pokud soubor v adresáři s programem neexistuje tak se v něm vytvoří. Po uložení slova do souboru je toto slovo, v případě hodnoty 0 v proměnné `ctenitxt`, vypsáno do konzole. Následně je proměnná `hlasky` kompletně přepsána do proměnné `asimilace` (již jsem zmiňoval na začátku kapitoly 3.4), protože v příštím cyklu bude slovo obsažené v proměnné `hlasky` zpracováváno. Dále je také důležité uložit do proměnné `x2` hodnotu `x`, ve které se nachází počet použitých pozic (délka slova) v proměnné `hlasky`, kterou musíme přesunout zároveň se samotným slovem.



Obr. 3.4: Vývojový diagram pro výpis na obrazovku a do souboru

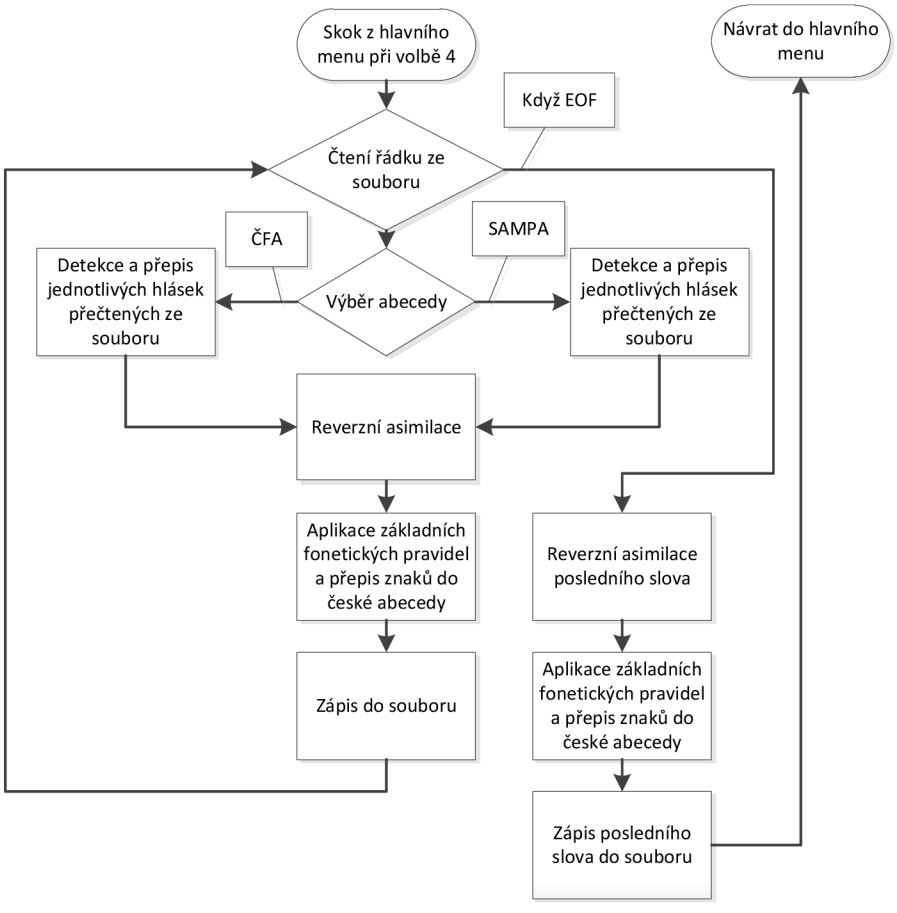
Aby bylo možné v příštím cyklu kontrolovat délku dalšího načteného slova, musíme vynulovat proměnnou `x`. Také je zde inkrementace proměnné `pruchod`, která tu je hlavně z důvodu prvního cyklu, kdy se neprovádí asimilace a celý její blok se přeskakuje. Hodnota v této proměnné se nuluje vždy při výběru možnosti 1, 2 nebo 3 v hlavním menu. Následuje jednoduchá funkce `if`, která v případě volby 3 v hlavním menu, uzavírá na základě výskytu návratové hodnoty EOF v proměnné `konecsouboru`, soubor určený pro čtení (již bylo zmiňováno na konci kapitoly 3.2). Zároveň je zde příkaz `break` pro ukončení programu a návrat do hlavní nabídky. Poslední funkce programu je ukončení zadávání slov pro překlad,

které je řešeno zapsáním třech středníku za poslední slovo. Jakmile se tato kombinace začne zpracovávat, `scanf` je uloží do proměnné `text`, kterou kontroluje funkce `if(strcmp(text,";;;") == 0)`, pokud je splněna, program pomocí příkazu `break` vyskočí z cyklu zpracování slov a vrátí se do hlavního menu. V případě, že podmínka splněna není, program se vrátí na začátek cyklu čtení, přečte další slovo nebo čeká na zápis dalšího slova.

3.6 Zpětný přepis z fonetického zápisu do původního textu

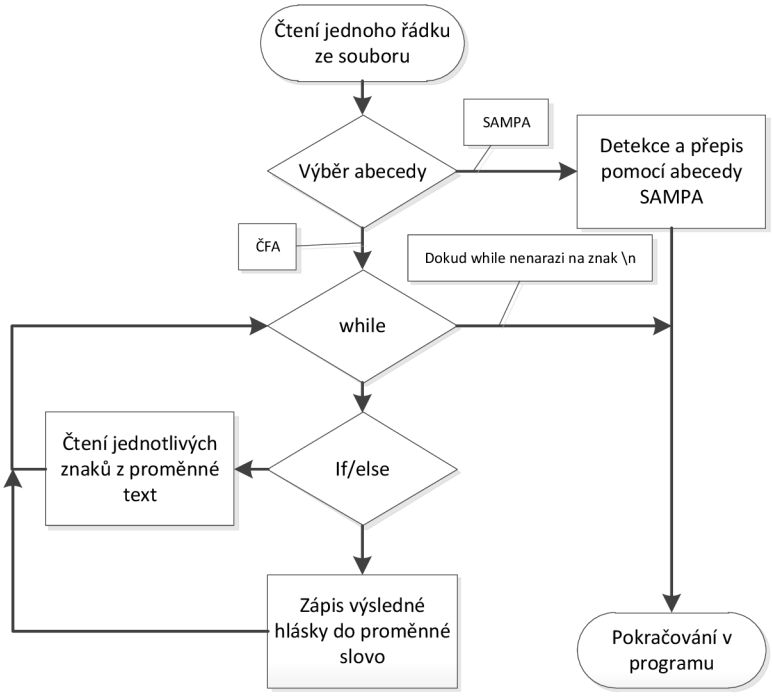
Z důvodu velké pravděpodobnosti výskytu chyby při ručním přepisu slova do fonetické podoby, jsem se rozhodl, že program bude schopný číst fonetický zápis pouze z textového souboru. V této části programu tedy používám soubory, které jsou výstupem první poloviny programu. Výstupem této části programu je přeložený text, uložený v kódové sadě 852, ve zvoleném textovém souboru. Pro úplnost ještě zopakuji, co vše uživatel volí při výběru čtvrté volby v hlavním menu. Uživatel nejdříve zadá jméno souboru, ze kterého chce číst. Následně je vyzván pro výběr abecedy (ČFA nebo SAMPA), ve které je přeložen zdrojový soubor. Následně je ještě vyzván pro zadání jména souboru, do kterého se má výsledný text uložit. Teď mohu přejít k samotnému popisu.

-Program je možné jednoduše rozdělit na tři hlavní části, první částí je čtení a detekce jednotlivých znaků ze souboru, v druhé části je provedena reverzní asimilace a v poslední části se současně s přepisem jednotlivých hlásek zpět na české znaky provádí zpětný přepis podle základních fonetických pravidel.



Obr. 3.5: Vývojový diagram pro zpětný přepis

-Program tvoří nekonečný cyklus `while`, ze kterého se na příslušném místě vyskakuje použitím příkazu `break`. Následuje funkce `if` s podmínkou:
`fgets(text,100,souborcteni) != NULL`, která přečte vždy jeden celý řádek z textového souboru, na kterém je (nehledě na zvolenou fonetickou abecedu) uloženo pouze jediné slovo. Zároveň program také hlídá, jestli se nedostal na konec souboru, v takovém případě skáče na funkci `else`, jejíž obsah budu popisovat ke konci této kapitoly. Uvnitř této funkce `if` se program nejdříve rozhoduje na základě uživatelsky zvolené abecedy, jakým způsobem se bude zpracovávat slovo uložené v proměnné `text`. Rozhodování se provádí, protože struktura zápisu slov do souboru je u abeced rozdílná, zatímco text uložený do souboru pomocí abecedy SAMPA je souvislý, text uložený pomocí abecedy ČFA má mezi jednotlivými hláskami mezeru, protože by při souvislém textu mohlo dojít ke špatné interpretaci slova (např. slovo *naaranžoval* by mohlo být interpretováno jako náranžoval). Zásluhou mezer u abecedy ČFA můžeme tedy jednoduše detekovat jednotlivé hlásky. V textu zpracovaném pomocí abecedy SAMPA je určování jednotlivých znaků složitější.
Nejdříve tedy popíšu čtení slov ze souboru, který byl přeložen abecedou ČFA. V proměnné `text` máme již načten celý řádek z textového sboru, ve kterém se nachází jedno slovo. Samotné čtení jednotlivých hlásek je uvnitř cyklu `while(text[i]!='\n')`, který se provádí dokud nedetekuje znak „\n“. Tento znak nám říká, že jsme se dostali na konec řádku a celé slovo bylo přečteno. Uvnitř tohoto cyklu se nachází funkce `if/else` s podmínkou `text[i]==' '`, tedy pokud se v proměnné `text` narazí na mezeru, provede se obsah funkce `if`, v opačném případě se provádí funkce `else`. Nejdříve popíšu, co se provádí ve funkci `else`, protože při čtení ze souboru se bude vždy provádět jako první (žádné slovo nezačíná mezerou). V proměnné `text` se pohybujeme po jednotlivých znacích pomocí proměnné `i`, která je vždy na konci cyklu `while` inkrementována o 1.

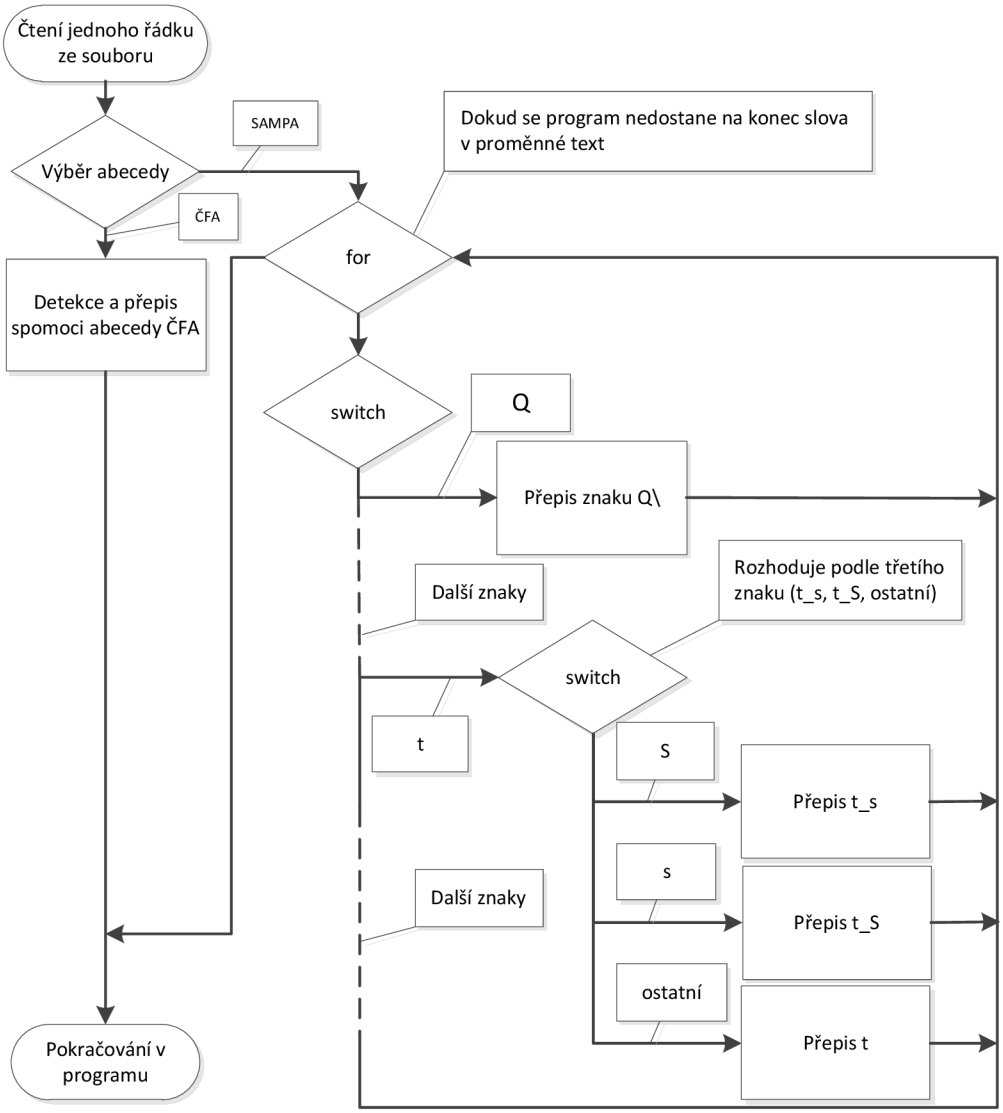


Obr. 3.6: Vývojový diagram pro detekci a přepis pomocí abecedy ČFA

V prvním cyklu je tedy `i` rovno 0 a z proměnné `text` se čte první znak hlásky, tzn. pokud první hláska slova je např. písmeno *ř* (které se přepíše pomocí abecedy ČFA na *rsh*) v prvním cyklu se čte písmeno *r*. Písmeno *r* se zapíše do proměnné `fonznak`, v druhém cyklu se do proměnné `fonznak` připíše písmeno *s* a ve třetím cyklu se připíše ještě písmeno *h*. Tím je v proměnné `fonznak` uložen celý řetězec jedné hlásky (*rsh*). Ve čtvrtém cyklu program přečte znak následující za hláskou *rsh*, což je mezera a program se dostává poprvé do funkce `if`. Zde se pomocí příkazu `strcpy(slovo[z],fonznak)` provede zápis hlásky uložené v proměnné `fonznak` do proměnné `slovo` na první pozici, protože v prvním cyklu je proměnná `z` rovna 0. Dále se ještě na tuto pozici připíše za samotnou hlásku mezera, aby se znaky obsažené v proměnné `slovo` shodovali se znaky uloženými v proměnné `cfa` případně `abeceda`. Po připsání mezery ještě následuje inkrementace hodnoty v proměnné `z` o 1, která určuje délku slova. V dalších cyklech jsou do proměnné `slovo` stejným způsobem připsány i další hlásky. Jakmile `while` narazí na znak „\n“, tak se cyklus dostal až na konec proměnné `text` a v proměnné `slovo` jsou tedy uloženy všechny hlásky slova přečteného z textového souboru. V proměnné `z` je dále uložena délka tohoto slova. Splněním podmínky se vyskočí z cyklu `while` a pokračuje se dále v programu.

Teď mohu přejít k popisu způsobu čtení hlásek ze souboru s textem, přeloženém v abecedě SAMPA. Budu předpokládat, že v proměnné `text` je již uložen jeden řádek se slovem z textového souboru. Při čtení znaků abecedy SAMPA z proměnné `text` je použit totožný cyklus, jako byl použit při určování základních fonetických pravidel v první polovině programu. Nejdříve se pomocí příkazu `strlen` změří délka slova v proměnné `text`. Tato hodnota se uloží do proměnné `i`, která určuje počet průchodů cyklem `for` a zároveň pomáhá, při určování konkrétního znaku v proměnné `text`. Následuje přepínač `switch`, který se přepíná podle písmena, které určuje výraz `text[strlen(text)-i]`(podrobný popis tohoto cyklu je v prvním odstavci kapitoly 3.3). Například při detekci písmena *i*, `switch` přeskočí na příslušný `case`, ve kterém se nachází funkce `if/else` s podmínkou `text[strlen(text)-i+1] == ':']`. Tedy v případě, že následující znak je dvojtečka se provede obsah funkce `if`, ve které se zapíše pomocí příkazu `slovo[z]=abeceda[7]` do proměnné `slovo` znak *i:* ze sedmé pozice proměnné `abeceda`. Následně se ještě inkrementuje o 1 hodnota v proměnné `z`, aby se příští detekovaný znak zapsal na další pozici v proměnné `slovo` a také se dekrementuje o 1 hodnota v proměnné `i`, která přeskočí čtení znaku dvojtečky v proměnné `text` v příštím cyklu. V případě, že následující znak za *i* není dvojtečka se do proměnné `slovo` zapíše samotné *i* příkazem `slovo[z]=abeceda[2]` a inkrementuje se hodnota v proměnné `z`. V případě detekce např. písmene *a* je určení znaku mírně složitější, protože se může jednat o hlásku *a*, *a:*, nebo *a_u*. Místo funkce `if` je zde `switch`, který přepíná na základě výrazu `text[strlen(text)-i+1]`, tedy podle znaku následujícím za znakem *a*, pokud je následující znak dvojtečka zapíše se do proměnné `slovo` hláska *a:*, o 1 se inkrementuje proměnná `z`, a také se o 1 dekrementuje proměnná `i` pro přeskočení čtení dvojtečky v příštím cyklu, jako v předchozím případě. V případě detekce podtržítka se zapíše do proměnné `slovo` hláska *a_u*, o 1 se inkrementuje proměnná `z` a o 2 se dekrementuje hodnota v proměnné `i`, protože chceme přeskočit čtení dvou znaku (znak

podtržítka a *u*) v proměnné `text`. V případě, že se nedetekuje ani jeden ze znaku, se do proměnné `slovo`, zapíše jen samotné *a*. Zápis ostatních hlásek do proměnné `slovo` je obdobný.



Obr. 3.7: Vývojový diagram pro detekci a přepis pomocí abecedy ČFA

Tímto je vysvětlena první hlavní část: čtení a detekce jednotlivých znaků. Dále budu popisovat druhou část programu, kde dochází k reverzní asimilaci.

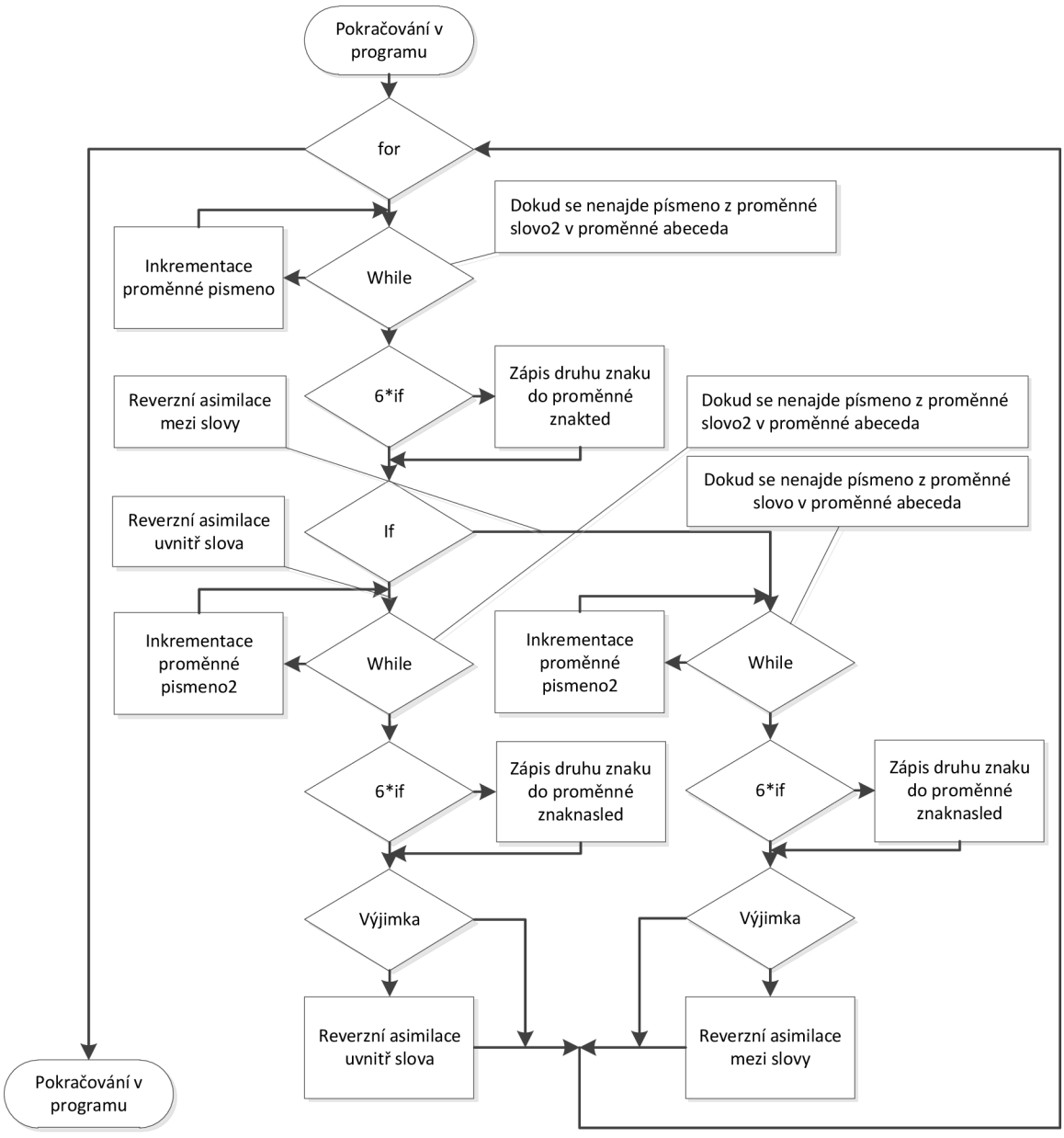
Při asimilaci v první polovině programu, kdy se přepisoval zadaný text do fonetického zápisu, se asimilace zpracovávala od zadu. Pravidlem při asimilaci je, že poslední písmeno souhláskové skupiny určuje znělost celé skupiny. Z tohoto pravidla jsem vycházel při zpracování reverzní asimilace. Reverzní asimilaci tedy provádím ve směru čtení (z leva doprava). Pokud narazím na skupinu po sobě jdoucích souhlásek, změním znělost všech souhlásek kromě posledního. Zde jsem ale narazil na problém, protože některá slova mohou obsahovat souhláskové skupiny, které mají stejný druh znělosti už v normální podobě. Jako příklad mohu uvést např. slovo *městský*, ve kterém se nachází souhlásková skupina *stsk*,

všechna souhlásky v této souhláskové skupině jsou neznělé párové souhlásky. Při asimilaci tedy nedošlo ke změně znělosti žádné souhlásky, ale při reverzní asimilaci se podle pravidla změní všechny souhlásky v souhláskové skupině kromě posledního a ze slova *městský* se stane slovo *mězdzký*. Jako další příklad mohu uvést slova *zkouška* a *český*, která se přepíšou do fonetické podoby jako *skouška* a *český*. V obou slovech se nachází souhlásková skupina *sk*, ale jen v jednom slově se má *sk* přepsat zpět na *zk*. Tento problém by se dal vyřešit, jen kdyby program spolupracoval se slovníkem. Provedení reverzní asimilace se hodně podobá zpracování asimilace v první polovině programu. Stejně jako při asimilaci se i v reverzní asimilaci pracuje až v druhém cyklu, kdy je ze souboru přečteno i následující slovo. K tomu je zde použita proměnná `pruchod`, která se inkrementuje o 1 na konci cyklu reverzní asimilace. Ke zpracování reverzní asimilace se program dostane, pokud je hodnota proměnné `pruchod` větší než nula. V prvním cyklu se slovo obsažené v proměnné `slovo` přesune do proměnné `slovo2`, zároveň se do proměnné `z2` uloží hodnota proměnné `z`, která určuje délku slova, ta se poté vynuluje. Pokud se nacházíme v druhém cyklu a v proměnných `slovo` a `slovo2` jsou uloženy dvě po sobě následující slova, program přechází k samotné reverzní asimilaci. Zpracovává se v cyklu `for`, který se provede tolikrát, kolik je hlásek ve zpracovávaném slově, to je určeno hodnotou v proměnné `z2`. Následuje určování druhu znaku (ZPK, NPK, JK,...), které se provádí od začátku slova. Pro ukládání druhu znaku se zde používají proměnné `znakted` a `znaknasled`. Do proměnné `znakted` se ukládá druh právě zpracovávaného znaku do proměnné `znaknasled` se ukládá druh následujícího znaku. Druh znaku se určuje stejným způsobem jako při asimilaci, tedy v cyklu `while` s podmínkou `strcmp(slovo2[i],abeceda[pismeno]) != 0`, se provádí inkrementace hodnoty v proměnné `pismeno`. Jakmile je splněna podmínka, v proměnné `pismeno` je uložena pozice písmena (v proměnné `abeceda`), které se shoduje s písmenem v proměnné `slovo2`.

Zásluhou vhodně rozvrženého pořadí hlásek v proměnné `abeceda`, lze určit, o jaký druh hlásky se jedná. Hodnoty 0-12 jsou samohlásky nebo dvojhlásky (V), hodnoty 13-23 jsou znělé párové souhlásky (ZPK), hodnoty 24-34 jsou neznělé párové souhlásky, hodnoty 35-42 jsou jedinečné souhlásky (JK), hodnoty 43 a 44 jsou funkční znaky (X) a hodnota 45 je pauza (P). Tato zkratka se zapíše do proměnné `znakted`. Stejným způsobem dojde k určení druhu následujícího znaku, akorát se místo proměnné `pismeno` používá proměnná `pismeno2` a výsledek se ukládá do proměnné `znaknasled`. Při zpracování posledního písmene program nečte následující písmeno, ale první písmeno následujícího slova, které může taky ovlivnit reverzní asimilaci. Provádí se to stejným způsobem jako určování druhu znaku následujícího písmene (používají se proměnné `pismeno2` a `znaknasled`) s tím rozdílem, že cyklus `while` má podmínku: `strcmp(slovo[0],abeceda[pismeno2])!=0`.

-Jakmile program zná druhy znaku dvou po sobě následujících hlásek, může hlásky zpracovat podle pravidel reverzní asimilace. Pro příklad uvedu jedno ze základních pravidel (které způsobují již zmíněnou chybu), funkce `if` má podmínku:
`strcmp(znakted,"NPK")== 0 && strcmp(znaknasled,"NPK") == 0`, tedy její obsah se provede, pokud dva po sobě následující znaky jsou oba neznělé. Toto pravidlo změní pomocí příkazu `strcpy(slovo2[i],abeceda[pismeno-11])`, právě

zpracovávaný znak z neznělého na znělý. Tedy například již zmíněné slovo *zkouška*, vypadá ve fonetickém zápisu jako *skouška*, jsou zde dvě souhláskové skupiny *sk* a *šk*. Obě souhláskové skupiny jsou zásluhou neznělého *k* na konci také neznělé. V proměnné `znakted` je uložena zkratka NPK (podle *s*), v proměnné `znaknasled` je také uložena zkratka NPK (podle *k*), tím se splnila podmínka a neznělé *s* je zaměněno pomocí výše zmíněného výrazu za znělé *z*. Ale zásluhou stejného pravidla je v souhláskové skupině *šk*, zaměněno neznělé písmeno *š* za znělé *ž*, a výsledné slovo, na kterém se provedla reverzní asimilace, je *zkoužka*, čímž se dostáváme k již zmíněné chybě při přepisu.



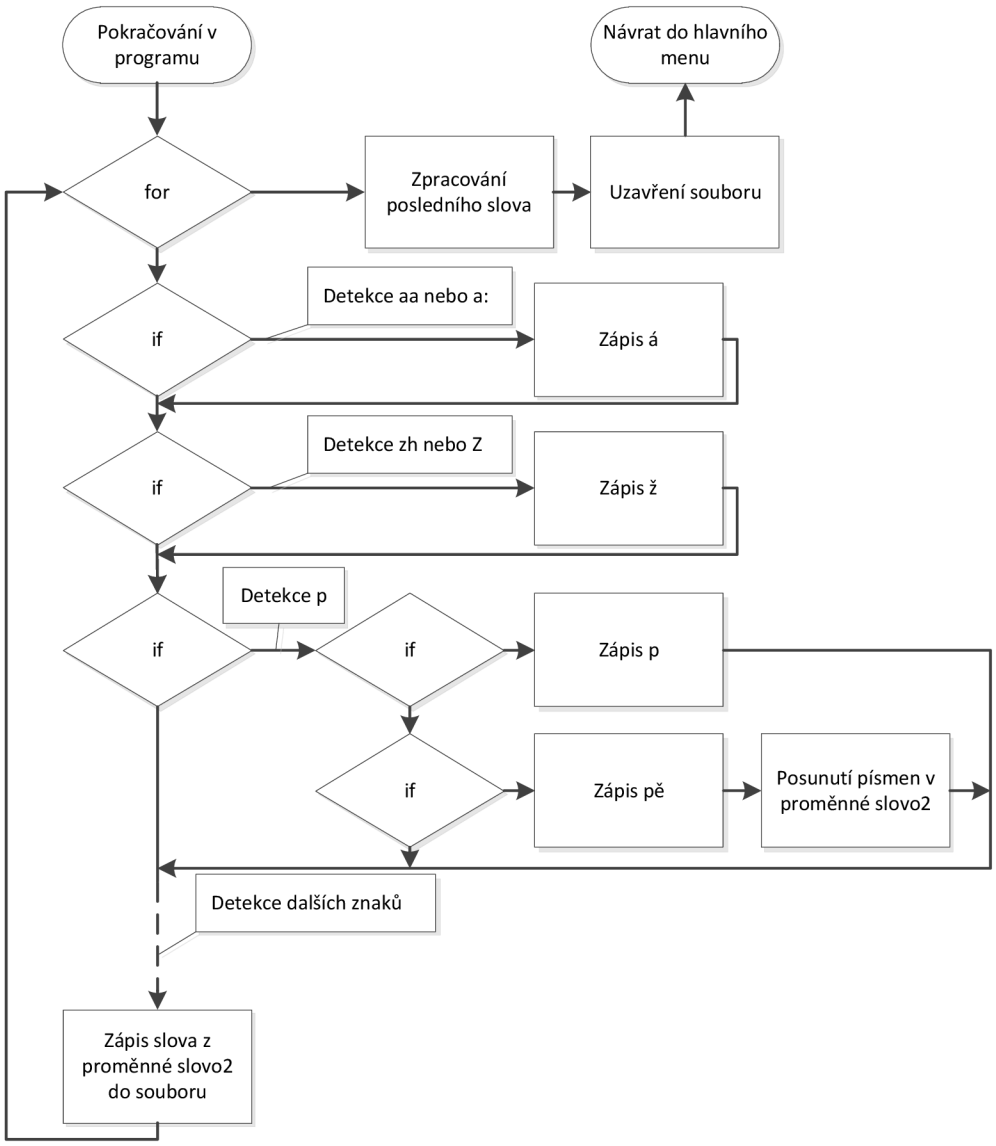
Obr. 3.8: Vývojový diagram pro reverzní asimilaci

Ostatní pravidla jsou řešena obdobně. Výjimky z reverzní asimilace jsou řešeny stejně jako při asimilaci, tedy například funkcí `if` s podmínkou `slovo2[i]!=abeceda[17] && z2!=1`, která řeší výjimku změny předložky *v* na *f*. Kontroluje, jestli se v proměnné `slovo2`

nenachází písmeno *v* a jestli zároveň není délka slova 1. Pokud se tak stane obsah funkce `if` je přeskočen a neprovede se reverzní asimilace. Ostatní výjimky z reverzní asimilace jsou řešeny obdobně. Tímto je popsána druhá hlavní část programu reverzní asimilace a můžu přejít k popisování poslední části programu, kde se současně s přepisem jednotlivých hlásek zpět na české znaky provádí zpětný přepis podle základních fonetických pravidel.

V poslední části programu po reverzní asimilaci se jednotlivé znaky v proměnné `slovo2` zaměňují za znaky české abecedy, které jsou uloženy v proměnné `ceska` (tabulka 3.1). Záměna znaků případně změna podle základních fonetických pravidel je prováděna v cyklu `for`, který se provede tolikrát, kolik je hlásek ve zpracovávaném slově (to je určeno hodnotou v proměnné `z2`). Pro každé písmeno je v cyklu jedna funkce `if`, jejíž obsah se provede, pokud je splněna podmínka. Znaky, které nejsou ovlivněny žádným ze základních pravidel fonetického přepisu (základní pravidla jsou např. přepis *y* na *i*, *bě* na *bje* nebo *di* na *ďi*), jsou jednoduše zaměněny za znak české abecedy. Např. pro písmeno *á* vypadá podmínka funkce `if` takto: `strcmp(slovo2[i],abeceda[5]) == 0`. Pokud se shoduje znak v proměnné `slovo2` se znakem *á* (*a:* nebo *aa*, podle abecedy) je splněna podmínka funkce `if`, a provede se její obsah, což je jen příkaz `strcpy(slovo2[i],ceska[5])`, tedy na příslušné místo se zapíše znak *á* z proměnné `ceska`. Změna písmen podle základních pravidel není o moc složitější. Např. při detekci písmena *ď* (*dj* nebo Ʌ\) se uvnitř příslušné funkce `if` nachází další čtyři funkce `if`, které v podmínce kontrolují znak následující za znakem *ď*. Za znakem mohou být znaky *i*, *í* a *e* (s *ď* dohromady znamenají *di*, *dí*, *dě*), pro každou možnost je zde příslušná funkce `if`, pokud není splněna ani jedna z podmínek zapíše se do proměnné `slovo2` znak *ď*. Pokud je např. jako následující znak detekován znak *i*, zapíše se pomocí výrazu: `strcpy(slovo2[i],ceska[14])` do proměnné `slovo` znak *d*, a následně se pomocí výrazu `strcpy(slovo2[i+1],ceska[2])`, zapíše na následující pozici v proměnné slov znak *i*. Následně se ještě inkrementuje hodnota v proměnné `i` o 1 aby se v příštím cyklu přeskočilo čtení písmena *i*. Poslední části přepisu základních pravidel je přepis spojení *bě*, *pě*, *vě*, *fě*, *mě*, které se přepisují jako *bje*, *pje*, *vje*, *fje*, *mňe*. Při přepisu do původní podoby dochází ke změně velikosti slova, protože se jedno z písmen ztratí. Pro příklad uvedu třeba slovo *pěvec*, které je přepsáno jako *pjevec*. Pokud program detekuje spojení *pje*, tak nejdříve do proměnné zapíše *p*, následně na další pozici zapíše znak *ě* a obsah proměnné vypadá takto: *pěevec*. Tím došlo ke zkrácení slova a je potřeba všechny písmena za písmenem *e* přesunout o jednu pozici zpět. Toho je dosaženo následujícím cyklem:
`for(int x = (i+2); x < (z2-1); x++)`. Do proměnné `x` se uloží hodnota proměnné `i` zvýšená o 2, tato hodnota slouží k určení, na jakou pozici zapíše (od jaké pozice má dojít k přesunu) výraz `strcpy(slovo2[x],slovo2[x+1])` následující písmeno. V našem případě se slovem *pěevec* se v prvním cyklu přepíše písmeno *v* na pozici písmena *e* , tedy slovo bude vypadat takto: *pěvvec*, v dalším cyklu se přesune *e* na pozici *v* (pěveec) atd. až se přesunou všechna písmena. Následně se ještě o 1 dekrementuje hodnota v proměnné `z2`, která určuje délku slova. Obdobně je řešeno zkrácení slova ve spojeních *bje*, *vje*, *fje*, *mňe*. Po přepisu všech znaků a aplikaci základních pravidel je už v proměnné `slovo2` výsledné slovo určené k zápisu do souboru. Po zápisu do souboru je přepsán obsah proměnné `slovo` do

proměnné `slovo2` současně s přepisem hodnoty z proměnné `z` do proměnné `z2`. Zde program inkrementuje hodnotu proměnné `pruchod` a vrací se zpět ke čtení dalšího slova ze zdrojového souboru. V případě, že program přečetl poslední slovo ze souboru tak se provede reverzní asimilace bez možnosti ovlivnění dalším slovem, aplikují se základní pravidla, slovo je přepsáno zpátky do české abecedy a uloženo do souboru, následně je cílový soubor uzavřen příkazem `fclose (souborcteni)` a program skočí zpět do hlavního menu.



Obr. 3.9: Vývojový diagram pro aplikaci základních fonetických pravidel a přepis znaků do české abecedy

4 Závěr

Během bakalářské práce jsem nastudoval metody zápisu zvukové podoby českého jazyka. Získané znalosti jsem následně použil, ke tvorbě programu v programovacím jazyce C. Program dokáže přeložit text napsaný do konzole nebo ho číst z textového souboru. Uživatel si může vybrat, do jaké fonetické abecedy chce zdrojový text přeložit, na výběr jsou abecedy ČFA a SAMPA. Výsledný přeložený text může být uložen do souboru s uživatelsky volitelným názvem a příponou.

-Dalším cílem bakalářské práce byl zpětný překlad z fonetického do normálního zápisu. Program je schopný z textového souboru přečíst text, aplikovat na něho pravidla zpětné asimilace, dále také základní fonetická pravidla, jednotlivé znaky přepsat zpět na znaky české abecedy a následně uložit výsledný text do textového souboru. Zpětný převod se ukázal jako mnohem složitější, neboť díky asimilaci výslovnosti fonetický přepis jednoznačně nedefinuje původní slovo. Např. fonetický zápis *skaakat* je přepisem slova *skákat*. Ale zápis *skouška* nevznikl přepisem *skouška* ale *zkouška*, neboť neznělé *k*, způsobilo asimilaci znělosti předchozího *z*. Pokud bychom ale asimilaci předpokládali i v předchozím případě, z přepisu *skaakat* bychom získali nesprávné slovo *zkákat*. V tomto případě by bylo možné správnou podobu slova nalézt pomocí automatické kontroly pravopisu nebo slovníku českých slov. Ještě složitější situace nastane u dvojice slov *zpráva* a *správa*, kde výslovnost obou slov je ve fonetické abecedě zapsána jako *spraava*, protože neznělé *p* způsobí asimilaci znělého *z*. Z fonetického zápisu tak není možné jednoznačně určit, které z obou slov bylo původně vysloveno. Člověk tyto nejednoznačnosti rozliší podle kontextu, ve kterém slovo slyší. V případě strojového překladu by v takové situaci nepomohla ani kontrola pravopisu. Kontrola výstupu zpětného přepisu pomocí automatické kontroly pravopisu je jednou z možností dalšího rozšiřování programu. Správný přepis cizích nebo přejatých slov, u kterých se vyskytuje velké množství různých výjimek, by mohla být vylepšena spoluprací s elektronickým slovníkem přejatých slov.

Literatura

[1] PSUTKA, Josef, et al. *Mluvíme s počítačem česky*. Praha : Academia, 2006. 746 s. ISBN 80-200-1309-1.

[2] PALKOVÁ, Zdena. *Fonetika a fonologie češtiny*. 1. vyd. Praha : Karolinum, 1994. 366 s. ISBN 80-706-6843-1.

[3] POKORNÁ, J.; VRÁNOVÁ, M. Přehled české výslovnosti. 1. vydání. Praha: Portál, 2007. 928 s. ISBN 978-80-7367-169-3

[4] HEROUT, Pavel. *Učebnice jazyka C. 1. díl*. IV. přepracované vydání. České Budějovice: Kopp, 2004. ISBN 80-7232-220-6.

[5] *Cplusplus.com* [online]. [cit. 2012-05-25]. C Language Library. Dostupné z: WWW: <http://www.cplusplus.com/reference/clibrary/>

[6] *Slova.oficialni.cz* [online]. [cit. 2012-05-25]. Česká slova - Seznam slov. Dostupné z WWW: <http://slova.oficialni.cz/>.

Seznam zkratek

ASCII - American Standard Code for Information Interchange
ČFA - Česká Fonetická Abeceda
EOF - End Of File
IPA - International Phonetic Alphabet
JK - Jedinečné souhlásky (Konsonanty)
JPZ - Jednoslabičné Předložky (končící znělou souhláskou)
NP - Neslabičné Předložky
NPK - Neznělé Párové souhlásky (Konsonanty)
P - Pauza
SAMPA - Speech Assessment Methods Phonetic Alphabet
V - samohlásky (Vokály) a dvojhlásky
X-SAMPA - eXtended Speech Assessment Methods Phonetic Alphabet
ZČFA - Zjednodušená Česká Fonetická Abeceda
ZPK - Znělé Párové souhlásky (Konsonanty)

Obsah CD

-Celá složka projektu - zde je uložená celá složka projektu pro Microsoft Visual Studio 2008

-El. verze práce - Fonetická transkripce českého jazyka.pdf (El. verze práce)

-Program - transkripce.exe (přeložený program)
- babicka.txt (testovací text pro volbu 3)
- test.txt (testovací fráze pro volby 1 a 2)

-Přeložené texty - babicka cfa.txt (text přeložený programem do abecedy ČFA)
- babicka sampa.txt (text přeložený programem do abecedy SAMPA)

-PSPad - Soubory programu PSPad (Program pro čtení a zápis v kódové sadě 852)

-Zdrojový soubor - transkripce.cpp (Zdrojový soubor programu)

-Návod.txt (Návod jak otestovat všechny funkce programu)