CRIiSTAL REFERENCE

Nel **CRIiSTAL reference** saranno descritti i programmi realizzati e la loro suddivisione nelle varie cartelle, evidenziandone gli aspetti funzionali. Per la struttura delle pagine Web si rimanda allo schema riportato nell'introduzione.

2.1 LA CARTELLA WEBROOT

All'interno della cartella troviamo la cartella contenente i dati del database, quella con i fonts utilizzati nel programma, la cartella contenente il modulo linguistico, la cartella contenente il modulo di sintesi e quella con i volumi della tesi in formato elettronico. Per questi ultimi si è già fatto cenno all'interno dell'interfaccia grafica presente nel **CRIiSTAL guide**. Esamineremo perciò in dettaglio, la cartella convertitore alfabeto e quella sintetizzatore. Si tenga presente che lavorando con pagine di tipo **html**, i codici che permettono l'elaborazione delle variabili ed il loro invio tramite protocollo **HTTP** sono "immersi" all'interno della pagina stessa e confinati all'interno di specifici tags che ne indicano la natura: abbiamo ad esempio di tags **<? ?>** per il **PHP** e quelli **<!--** **-->** per il **Javascript**. Occorrerà prestare attenzione a questi aspetti per comprendere le funzioni svolte dalle singole righe di codice.

2.2 CONVERTITORE ALFABETO

All'interno di questa cartella ci sono la cartella convertitore italiano, convertitore spagnolo, correzione, funzioni, immagini, file **UTF** e le pagine di immissione dati e di scelta lingua. Le ultime due cartelle contengono semplicemente le immagini utilizzate all'interno di tutto il progetto ed i files di testo di tipo unicode generati durante le elaborazioni.

La pagina di sceltalingua è una pagina di tipo **HTML** e non presenta particolari aspetti da sottolineare. Va invece spiegato l'invio dei dati da parte della pagina di immissione al programma seguente.

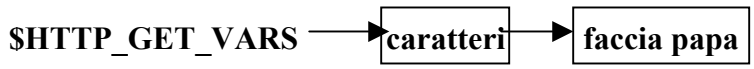
L'inserimento di una variabile in una pagina Web è possibile tramite un **input text** di una **form HTML**. A questo **input text** è associato un nome. Il passaggio di variabili fra le varie pagine è supportato dal protocollo **http** e può essere effettuato in due modalità

GET con visualizzazione immediata sulla barra di navigazione delle varie variabili

POST con mascheramento delle variabili passate

A seconda della modalità scelta, nell'array associativo **$HTTP_MODSCELTA_VARS** troveremo nell'elemento contrassegnato dal nome usato nella input text, il valore immesso.

Nel nostro caso il nome identificativo dell'**input text** è **caratteri** e la modalità di trasferimento dati è la **get**. Inserendo la stringa "*faccia papa*" avremo nel vettore **$HTTP_GET_VARS** nell'elemento con parola chiave **caratteri** il valore "*faccia papa*".



Per inviare effettivamente questi dati alla pagina desiderata, dobbiamo specificare nell'attributo **button** il percorso per raggiungere il file destinatario, cioè la sua **URL**, che nel nostro caso è il **convertitore_italiano**. Al momento dell'invio i dati saranno scritti nel vettore come descritto in precedenza e saranno disponibili al convertitore per l'elaborazione.

2.2.1 LA CARTELLA CONVERTITORE ITALIANO

In questa cartella troviamo il programma principale di elaborazione e quella ausiliaria di scelta degli omografi. Il programma convertitore italiano ha nella sua intestazione come prima dichiarazione quella dell'avvio di una sessione: questa istruzione deve precedere qualsiasi altra all'interno del programma. Vengono poi specificati i nomi delle variabili di sessione utilizzate. Le variabili di sessione

sono variabili visibili ed accessibile da tutti programmi relativi alla medesima sessione di lavoro e sono perciò editabili da pagine diverse senza bisogno di scambiare dati fra le stesse. Va comunque limitato l'uso di questo tipo di variabili per evitare sia una più difficile lettura del codice, sia una possibile instabilità del programma. Ciò si può verificare scorrendo le pagine **Web** a ritroso anziché nella direzione indicata dai links, che assicurano invece il corretto aggiornamento delle variabili.

Le variabili relative all'autenticazione, **username** e **password** servono semplicemente ad abilitare le funzioni protette mentre le altre sono utilizzate per mantenere dati comuni a più pagine, o come semafori. Queste ultime vengono asserite ad ogni iterazione per poter eseguire correttamente il ciclo successivo.

Troviamo poi una lista di istruzioni di tipo **include**, che caricano in memoria tutte le funzioni richiamate dal programma principale o dalle funzioni in esso contenute. Si noti che occorre specificare l'indirizzo relativo all'interno del comando stesso. L'uso di **URL** relativi anziché assoluti dà il vantaggio di poter installare in altri computer tutto il pacchetto software senza dover riscrivere alcun indirizzo: occorrerà solamente indicare l'indirizzo contenente tutti gli applicativi al **Web server**.

Il programma suddivide la stringa inserita, tramite la funzione **explode**, nelle parole componenti che verranno caricate in un vettore. Si esegue quindi un ciclo per l'elaborazione di ogni singola parola controllando dapprima se la parola in questione è un omografo tramite la funzione **omografo**: nel caso dia esito positivo, viene visualizzata una tabella di scelta contenente al centro la parola immessa ed ai lati le due possibili scelte, evidenziandone la selezione tramite delle icone. Tramite il bottone di invio, i dati vengono inviate alla pagina scegliomografi, in cui vengono riepilogate le scelte effettuate. Indipendentemente dall'elaborazione seguita la parola viene poi precodificata con la funzione **cambia simboli** e vengono create tre stringhe: una codificata, un sillabata ed una utilizzata per successive elaborazioni. Ogni stringa sarà inviata alla relativa pagina tramite un proprio **button** dedicato.

Il test all'inizio programma è necessario in quanto la pagina **scegli omografi** invia la sua elaborazione nuovamente al programma convertitore e occorre tenere memorie del passaggio dei dati per avere un corretto funzionamento dello stesso. Ad esempio inserendo la parola "*accetta"*, nel primo giro il programma controlla se è un omografo e avendo esito positivo visualizza la tabella di scelta. Il risultato viene inviato nuovamente alla pagina del convertitore ma questa volta l'elaborazione deve proseguire, in quanto la scelta è stata già effettuata, altrimenti si creerebbe un loop. Inoltre se la parola è stata elaborata dalla funzione **omografo**, è superfluo elaborarla con la funzione **suffissi** e le altre funzioni, in quanto è stata già determinata la posizione dell'accento.

Queste due esigenze vengono ottemperate con l'utilizzo di due semafori realizzati nello specifico con le variabili di sessione **$vai** e **$omo**. Se **$vai** è vuota o vale zero occorre eseguire la prima parte del programma in quanto si tratta della prima iterazione. Nel caso si riscontrino omografi, nella pagina **scegliomografi** ambedue le variabili sono settate per consentire la prosecuzione dell'elaborazione nella pagina chiamante, altrimenti si setta la sola variabile **$vai**. La restante parte di programma, se trova settata la variabile **$omo** esegue solo la conversione di alfabeto con la funzione **cambiasimboli**, altrimenti svolge tutti i controlli previsti. Al termine di un ciclo relativo ad una parola, le variabili sono resettate. All'interno della pagina **scegliomografi**, la parola eventualmente scelta viene sostituita a quella immessa, inserendola nella posizione occupata nel vettore creato all'inizio del programma. Si creano, inoltre, anche le relative stringhe convertite con e senza accento. All'interno dei vettori **$vettoreomo** e **$duomo**, si inseriscono degli zeri all'interno di tutte le posizioni che non sono occupate da omografi. Per questo motivo **$vettoredistringhe**, **$vettoreomo** e **$dueomo** sono state specificate come variabili di sessione, cosicché scandendo il vettore, si elaborano solo le parole i cui indici non corrispondono a parole già esaminate nella procedura descritta in precedenza.

2.2.2 LA CARTELLA CONVERTITORE SPAGNOLO

La cartella in esame contiene al suo interno il programma principale. La struttura del programma principale è simile a quella del programma di italiano mancando però il controllo per gli omografi ed i suffissi. La parola immessa viene elaborata con la sola funzione codifica e sillabata utilizzando le funzioni specifiche che implementano le regole grammaticali spagnole. Infine viene assegnato l'accento. Le funzioni richiamate sono contenute nella cartella funzioni spagnole.

2.3 LA CARTELLA FUNZIONI

La cartella funzioni contiene quattro cartelle, due contenenti le funzioni specifiche per le lingua implementate, l'italiano e lo spagnolo, le altre due contenenti funzioni comuni a tutti programmi, come quelle per la sillabazione nella cartella funzioni sillabe e quelle per l'accesso ai database all'interno della cartella funzioni ausiliarie. Elenchiamo le funzioni nell'ordine in cui vengono invocate durante l'esecuzione del programma.

2.3.1 LA CARTELLA FUNZIONI ITALIANO

Funzione omografo

La funzione **omografo** riceve dal programma principale una stringa e controlla la sua presenza nella relativa tabella tramite la funzione di accessoadb. In caso affermativo la funzione restituisce al programma principale le parole da scegliere all'interno di un vettore altrimenti ritorna zero. Il programma principale se riceve il vettore omografo non vuoto, visualizza in una form le parole e tramite un bottone invia la scelta alla pagina **scegliomografi**

Funzione suffissi

Nel caso che la parola trattata non sia un omografo, il programma invoca la funzione **suffissi** che controlla la presenza nella stringa di particelle contenute nella tabella **prefissisuffissi** dette prefissi, che possono cioè, trovarsi in testa alla parola. La tabella viene scandita riga dopo riga: nel caso la ricerca dia esito negativo la parola è passata direttamente alle funzioni successive. Nel caso invece venga trovato almeno un prefisso, questo verrà separato dalla parola e le due parti subiranno un diverso trattamento: il primo verrà sillabato e convertito nell'alfabeto fonetico mentre la parte rimanente oltre ad essere sillabata e convertita verrà inviata alla procedura di predizione dell'accento. Per evitare situazioni ambigue sia effettua un controllo preventivo della lunghezza della parola che deve essere superiore a quella del prefisso individuato. In seguito, nel caso la parola presenti più di tre sillabe, viene effettuata la procedura suddetta. Questo ulteriore passaggio è richiesto poiché la funzione che predice l'accento può elaborare senza ambiguità fino ai trisillabi. La funzione restituisce al programma chiamante un vettore di due elementi, il primo contenente la parola sillabata senza accento ed il secondo contenente la parola sillabata con l'accento.

Funzioni convertitore

Sia in **omografi** che in **suffissi** vengono richiamate, in quest'ordine, le seguenti funzioni:

1) **codifica2-codifica**

2) **esamina**

3) **converti**

4) **accento**

Codifica2-codifica

Codifica2 riceve le parole da elaborare e tramite la funzione **controllaapertechiuse** verifica la presenza di terminazioni che contengano per loro natura delle e/o aperte o chiuse: individuarne la presenza permette di semplificare l'elaborazione successiva per determinare la posizione dell'accento nella stringa. Questa operazione è realizzata sempre tramite l'uso di espressioni regolari che, individuata la particolare sequenza di grafemi , sostituisce alla vocale individuata, la sua versione opportunamente accentata. Ad esempio, se si individua la sequenza "*ense*" in coda alla parola, la si sostituirà con la sequenza "*ènse*".

Indipendentemente dall'esito di questo controllo, la parola è passata alla funzione **codifica** che associa ad esse una stringa, che chiameremo stringa immagine, in cui è indicato in ogni posizione il tipo di grafema individuato, consonantico o vocalico. Vediamo un esempio:

sintesi articolatoria

cvccvcv vcvcvcvcvcvv

L'operazione è realizzata con l'uso di espressioni regolari che riconoscono l'appartenenza o meno del simbolo ad una certa categoria: scegliendo le vocali per semplicità, se il simbolo appartiene a questo insieme sarà inserito il simbolo "**v**" viceversa il simbolo "**c**". Sempre con le espressioni regolari si inserisce un simbolo separatore "/" in ogni transizione (cv vc) individuata. Se in coda trova suffissi che attirano l'accento, vengono separate le vocali interessate: ad esempio *grafia* subirà una divisione tra la "**i"** e la "**a"** ottenendo *grafi/a*. Utilizzando un contatore, si inseriscono i simboli di separazione nella parola originaria, scandendola parallelamente alla stringa immagine. Otterremo così una stringa in cui sono separate le vocali, le consonanti e le eventuali vocali che è già possibile separare. Ad esempio per parola biografia si avrà:

biografia ---> cvvccvcvv ---> c/vv/cc/v/c/v/v ---> b/io/gr/a/f/i/a

Tramite la funzione **explode** si ottiene il vettore

0) **b**
1) **io**
2) **gr**
3) **a**
4) **f**
5) **i**
6) **a**

che verrà passato alla funzione **esamina**.

Esamina

La funzione **esamina** riceve il vettore dalla funzione **codifica** e ne analizza in sequenza ogni elemento, ognuno dei quali viene processato da un blocco diverso a seconda che contenga consonanti o vocali. In ogni blocco troviamo un costrutto **switch**, le cui parti sono attivate utilizzando come indice la lunghezza dell'elemento esaminato. Questa procedura serve a verificare se i grafemi di ogni elemento debbano essere divisi oppure no per realizzare una corretta sillabazione. I segni di separazione usati nei due casi sono diversi: si utilizza il simbolo "/" per le vocali ed il simbolo "**-:**". Nel blocco per le consonanti troviamo un costrutto **switch** con due opzioni: per elementi formati da due grafemi e per elementi con un numero di grafemi maggiore di due. Ogni elemento viene analizzato con le funzioni che implementano le regole di sillabazione della lingua scelta: se una sola di questa dà esito positivo, viene inserito il simbolo di separazione tramite la chiamata della funzione **spezza**. Se l'elemento è formato da una sola consonante non viene inserito nessun elemento. A seconda del blocco chiamante, la funzione inserirà il corretto simbolo separatore per le vocali o per le consonanti. Per le vocali, anziché utilizzare le regole di sillabazione, occorrerà fare delle verifiche più articolate. Qui abbiamo un costrutto **switch** con sei opzioni, tante quante il numero massimo di vocali consecutive eventualmente presenti in un elemento.

Nel caso di una vocale singola non ci sono problemi e si inserisce semplicemente il simbolo separatore.

Negli altri casi occorre stabilire se le vocali presentano iato oppure no. In base alle proprietà desunte dal **DISC 95**, i casi di quattro, cinque e sei vocali consecutive, presentano una elaborazione certa. Diversamente accade nei casi di due e tre.

Nel caso di tre, eliminando i trittonghi notevoli come ad esempio **iuo**, occorre verificare dove si realizzi l'eventuale distacco tra le vocali. Si possono avere infatti tre casi: presenza di trittongo, dittongo seguito da vocale e viceversa, tre vocali separate di seguito o vocali usate come segni diacritici.

Nel caso due invece occorre capire se si ha iato, dittongo, vocali separate o segni diacritici. Il caso della "**i"** usata come segno diacritico è relativo al contesto ed è quindi univocamente determinabile. Nel caso che le vocali adiacenti siano "**a","e","o",** devono essere sempre separate e quindi anche in questo caso non si ha ambiguità. Diversa invece è la situazione nel caso che una delle due vocali sia **"i"** o **"u"** poiché si potrebbe avere sia iato che dittongo. Non esistendo regole a riguardo, se non nel caso di indicazione esplicita dell'accento, occorre inserire in una tabella le informazioni necessarie per una corretta elaborazione della coppia in esame. Per questo motivo lo spagnolo, invece, non è ambiguo, perché gli eventuali iati sono indicati obbligatoriamente, come la vocale portatrice di accento.

La consultazione della tabella **"iato"** avviene nel momento in cui si riscontra il primo elemento del vettore con due o più vocali che potrebbero essere separate, verificando se la parola immessa è presente al suo interno. In caso affermativo, cioè la parola contiene uno iato, dal campo posizione si ricavano le coordinate dell'elemento in cui è presente il distacco delle vocali e se ne fa il confronto con l'indice dell'elemento in esame: questo per evitare ulteriori accessi a db nel caso la parola presenti più coppie di vocali al suo interno. Ad esempio per la parola **miagolio** si ha:

0) m
1) ia
2) g
3) o
4) l
5) io

e sia nell'elemento 1) che nell'elemento 6) si interrogherebbe la tabella per ricavarne informazioni.

Invece i dati vengono caricati direttamente alla prima occorrenza riscontrata ed utilizzati nell'elemento corrispondente, migliorando l'efficienza e la velocità del programma. Oltretutto, se non si specificasse la posizione dell'elemento, si avrebbe ambiguità in quale dei due elementi deve avvenire il distacco.

Individuato l'elemento contenente le vocali da separare si invoca la funzione **spezza**. A questo punto, utilizzando l'esempio precedente avremo la seguente situazione:

0) m
1) ia/
2) g
3) o/
4) l
5) i/o

Tramite la funzione **implode**, utilizzando nessun simbolo di implosione, si ottiene la stringa mia/go/li/o. Con una successiva operazione di **explode** utilizzando però come separatore il simbolo **"/"** avremo:

0) mia
1) go
2) li
3) o

La parola è così correttamente sillabata. Nel caso si incontrino due o più consonanti come in *incanto* avremo:

0) i
1) nc
2) a
3) nt
4) o

Inserendo i simboli separatori si avrà:

0) i/
1) n-:c
2) a/
3) n-:t
4) o/

Tramite la funzione **implode**, utilizzando nessun simbolo di implosione, si ottiene la stringa: **i/n-:ca/n-:to/.** Con una successiva operazione di **explode** utilizzando però come separatore il simbolo **":"** avremo:

0) **i/n-**
1) **ca/n-**
2) **to/**

Eliminando i simboli **"-"** e spostando in coda i simboli **"/",** si ottengono in ogni elemento del vettore le varie sillabe. L'ultimo controllo effettuato dalla funzione verifica se l'ultima lettera è una consonante, nel qual caso la annette alla penultima sillaba e la rimuove dal vettore con la funzione **array_pop**. La penultima sillaba diventa perciò l'ultima.

Converti

In questa funzione sono implementate le regole di trascrizione fonetica esposte nel capitolo I del Volume I. Per motivi sintattici legati al linguaggio di programmazione non è stato possibile utilizzare direttamente tutti i caratteri fonetici standard ma si sono utilizzati dei simboli di appoggio che verranno correttamente sostituiti dalla funzione **cambiasimboli** in coda al programma principale. Per i gruppi **gn** e **gl** sono state inserite le parole che prevedono al posto della pronuncia palatale sonora nasale (*pugno*, *ragno*) e laterale continua palatale (*foglia*, *paglia*), la pronuncia della occlusiva velare sonora (*Wagner*, *anglicano*).

La trascrizione dei grafemi **"e"**, **"o"**, **"s"**, **"z"**, non è univoca in quanto ogni simbolo può corrispondere a due diversi fonemi.

Per le **"s"** e le **"z"**, integrando l'algoritmo riportato nell'articolo di Rodolfo del Monte sulla prosodia del parlato italiano con le regole di pronuncia del **D.O.P.**, si è ottenuto uno schema di analisi unico per entrambi i grafemi, riportato di seguito.

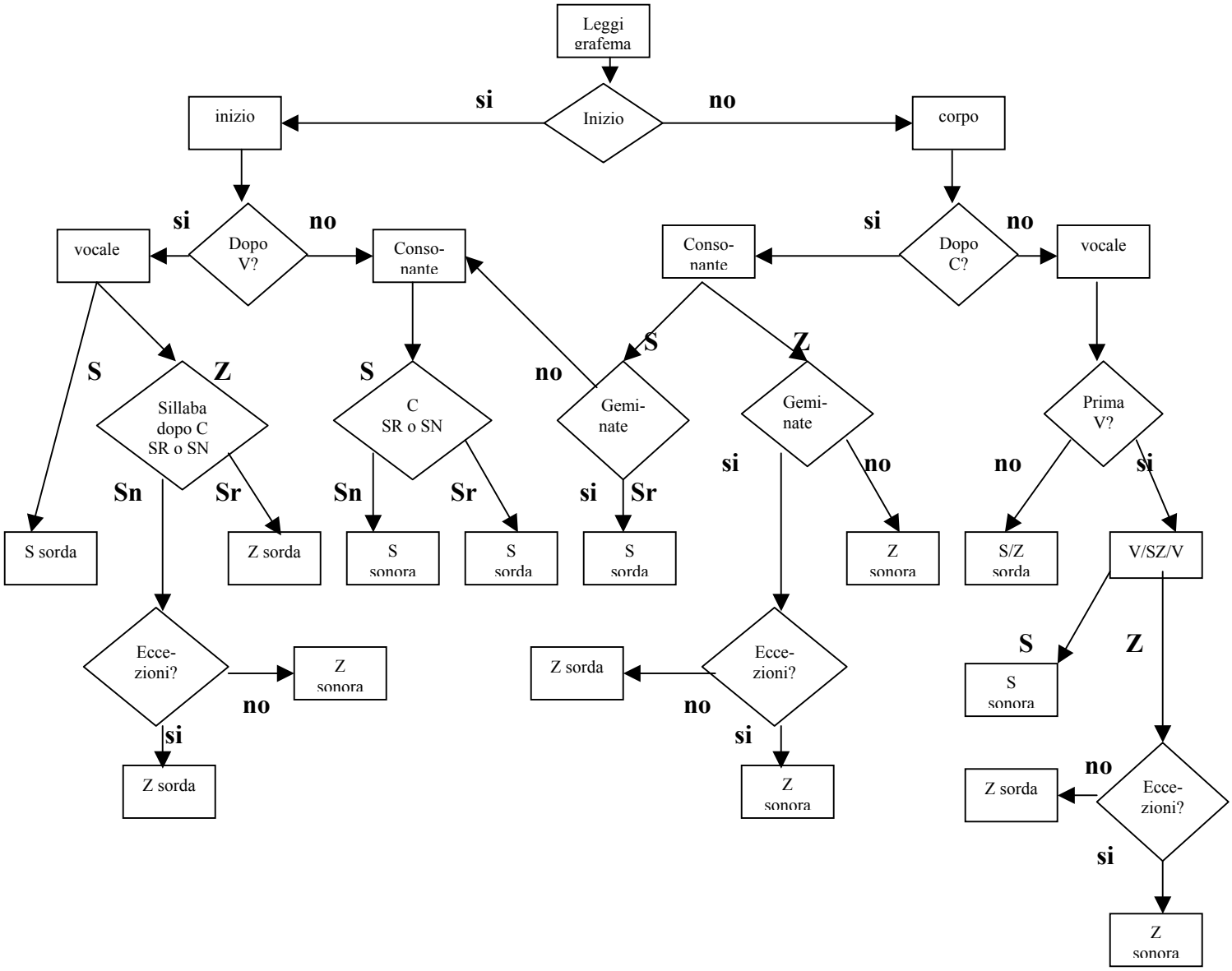


Figura 2.1 Algoritmo per associazione grafema – fonema per i fonemi "s" e "z"

Con il simbolo **"c"** si intende consonante, con **"v"** vocale, con **"Sr"** tipo sordo e con **"Sn"** tipo sonoro.

Per i grafemi **"e"** ed **"o"**, sono state invece utilizzate esclusivamente le regole riportate nel **D.O.P.** riguardanti le proprietà di alcuni suffissi, o combinazioni di grafemi, di presentare uno specifico fonema tra due possibili. In questa sede sono stati inseriti i controlli non implementati nella funzione **controllaapertechiuse**. Sia per le e/o che per le s/z, la verifica di eccezioni alle procedure scritte è effettuata con un accesso alla relativa tabella nel db, ottenendo come risposta le coordinate ed il tipo di lemento da modificare.

Tutte le conversioni di grafemi , vengono effettuate al solito con le espressioni regolari che, se si riscontrano una particolare sequenza, la sostituiscono con la relativa rappresentazione fonetica. Nei casi in cui ci possano essere più simboli da modificare della stessa natura, si troverà la regexp **preg_match_all,** che permette di memorizzare in un array tutte le occorrenze riscontrate. Ad esempio nella parola **caciocavallo**, vi sono tre **"c"**, che corrispondono però a due fonemi diverse: la prima e la terza sono c dure, mentre la seconda è una c dolce. Ognuna di esse dovrà essere convertita tenendo conto del grafema seguente ed occorre quindi una elaborazione individuale. Tramite un ciclo, scandendo ogni elemento dell'array, è possibile sostituire ogni singolo grafema senza modifiche indesiderate agli altri.

Accento

Questa è l'ultima funzione che incontriamo ed elabora il vettore fornito da **converti**. In essa sono implementate tutte le regole fonologiche esposte nel primo capitolo, effettuando controlli, se necessario, sui vari contesti presenti nella parola. L'elaborazione effettuata da qui in poi a livello di fonemi e non più di grafemi: ecco la necessità di far precedere la funzione **converti** a quella **accento**. Per le parole tronche ed i bisillabi, abbiamo visto che non ci sono problemi di ambiguità nell'assegnazione dell'accento. Viceversa accade nei trisillabi ed i polisillabi in genere.

Le istruzioni in testa alla funzione verificano la presenza di proprietà notevoli, eventualmente messe in risalto dalle funzioni precedenti, come la presenza di iati oppure e/o aperte/chiuse, che permettano di assegnare in maniera univoca l'accento.

Nel corso della stesura del programma si è evidenziata una curiosa circostanza: la simbologia utilizzata per indicare un contesto fonologico è praticamente uguale alla sintassi informatica delle espressioni regolari che lo implementano.

Contesto fonetico	*/t d l n m/*
Espressione regolare	*preg_match("/[t d l m n]/")*

Con questa notazione si intende "se è presente all'interno della stringa qualsiasi carattere fra *t,d,l,m,n"*.

La struttura della funzione è basata fondamentalmente sul costrutto **switch**, che utilizza come parametro di scelta il numero di sillabe riscontrato nella parola in esame. Sorvolando sul caso di una e due sillabe, che, come più volte detto non presentano ambiguità, analizziamo direttamente il caso dei trisillabi. In esso troviamo i due gruppi di regole principali relativi alla presenza o meno di consonanti geminate all'interno della penultima sillaba. Per far questo verranno memorizzate all'interno di apposite variabili l'ultimo ed il penultimo elemento del. Come si nota, e come sottolineato nel Volume I, le operazioni risultano tra loro disgiunte ed effettuando il solo controllo relativo alla presenza o meno della parola nella tabella delle eccezioni, l'assegnazione dell'accento risulterà univoca. I polisillabi vengono esaminati nel caso > 3, verificando la presenza di suffissi oppure di verbi aventi particelle enclitiche. Nel caso queste ultime vengano riscontrate, le si eliminano temporaneamente dalla parola e si richiama la funzione utilizzando come argomento la parte rimanente. Assegnata la posizione dell'accento, si ricostituisce la parola unendo la particella eliminata in precedenza.

Se la parola non rientra in nessuno dei casi implementati, viene automaticamente assegnata l'accentazione piana.

La procedura, purtroppo, richiede sempre un accesso al database: ciò è inevitabile sia per la natura del problema, sia per creare un meccanismo di correzione automatica.

Con il tempo comunque, si potranno evidenziare eventuali regole fonologiche sui termini inseriti, per implementare regole fonologiche simili a quelle usate nel caso dei trisillabi. Purtroppo i dati in nostro possesso sono ancora limitate per effettuare questo tipo di lavoro in maniera generale: si sono comunque trovate due regole inserite nella procedura dei polisillabi per le parole terminanti in *gnolo* e *colo*.

2.3.2 LA CARTELLA FUNZIONI AUSILIARIE

Funzione verbo

Questa funzione permette di verificare se la parola in esame sia un verbo avente come caratteristica gli argomenti inseriti nella chiamata della funzione relativamente alle persone, i tempi ed i modi. Attualmente questa funzione è in grado di operare correttamente solo con i verbi regolari. È possibile inserire sia un valore singolo che un intervallo di essi. Inserendo nel campo **“persone”** il valore **“2-5”** si controlleranno tutte le persone dalla seconda alla quinta mentre con l'espressione **“2&5”** saranno controllate solo la seconda e la quinta.

Per svolgere l’operazione si usa un array associativo avente nei campi chiavi i tre tipi di coniugazioni **“are”**,**”ere”**,**”ire”**. Gli argomenti immessi sono utilizzati come indici in costrutti **switch** annidati, consentendo di caricare nell’array le desinenze relative ad ogni coniugazione. Nel caso di immissione di intervalli, si esegue un ciclo per ogni valore da caricare. Per ogni chiave si verifica se vi sia la presenza nella parola immessa di almeno una delle desinenze caricate. In caso di esito positivo, la desinenze viene sostituita dalla chiave relativa tramite Una espressione regolare, ottenendo il verbo nella coniugazione all'infinito. Il verbo viene cercato nella tabella categorie aventi il tipo contrassegnato con **"v-inf"**. Se l’accesso va a buon fine, la funzione restituisce il verbo all’infinito al programma chiamante, viceversa zero. Questo approccio permette di ridurre drasticamente il numero delle voci verbali all'interno della tabella, evitando di memorizzare sei persone per ogni modo, per ogni tempo, per ogni verbo.

Funzione enclitico

Questa funzione controlla se in coda alla parola in esame siano presenti particelle enclitiche fino a tre livelli, utilizzando iterativamente delle espressioni regolari: si potrebbero infatti avere casi come fabbricalo, fabbricamelo e fabbricamicelo. Il controlla è doppio poiché si può avere anche il caso di verbi all’infinito come fabbricarlo o fabbricarcelo. Individuate le particelle, si eliminano e si richiama la funzione verbo con la restante parte della parola effettuando il controllo su tutte le persone, tutti

tempi e tutti i modi possibili. Nel caso la parola fosse un infinito, verrebbe usata direttamente per accedere alla tabella relativa. In caso di successo la funzione restituisce un vettore, contenente il verbo all'infinito e l'enclitico individuato, viceversa un vettore zero.

Funzione accessoadb

La funzione permette alle varie parti del programma in cui è necessario, di interrogare le tabelle per verificare la presenza o le proprietà di certe parole. La funzione utilizza come argomenti la parola da cercare e la tabella che la dovrebbe contenere. La prima operazione svolta è la connessione al db per verificarne la presenza o come si dice in gergo, se il db è **"tirato su"**. In caso di risposta positiva, viene assegnato un identificatore di connessione tramite il quale è possibile inviare una **"query"**, cioè una interrogazione in linguaggio **SQL** (structured query language), relativa all'operazione da effettuare. Per vedere le istruzioni **SQL** usate e la loro sintassi o le istruzioni **PHP** che fanno uso di esse, si rimanda ai relativi manuali o ai brevi esempi nel Volume I. La ricerca che viene effettuata prosegue su tutte le righe presenti nella tabella fino a che non viene riscontrata la parola cercata, caso in cui la funzione ritorna uno. Se la scansione avviene su tutta la tabella fino alla fine senza successo ritorna zero.

Funzione accessoadb2

Questa funzione permette di aggiornare il database durante la procedura di correzione ed a seconda della caratteristica da correggere selezionata, l'elaborazione viene indirizzata tramite il costrutto **switch**: al suo interno viene effettuata una **"query"** relativa alla modalità scelta. Nel caso di semplice correzione ad esempio, sarà necessario inserire il dato in esame previa cancellazione all'interno della tabella. Nel caso di omografo invece, il dato dovrà eventualmente essere eliminato da tutte le tabelle ed inserito unicamente in quella degli omografi.

2.3.3 LA CARTELLA FUNZIONI SILLABE

Funzione digrammi

Alla funzione viene passata la variabile **$app** dal programma chiamante e nel caso trovi la presenza dei digrammi **"gh", "gl", "ch", "cl", "gn"** ritorna 1.

Funzione gruppi

Alla funzione viene passata la variabile **$app** dal programma chiamante e se trova una delle combinazioni fra i grafemi contenuti nella prima parentesi quadra e la seconda, ad esempio **“bl”, “br”, “dl”, “dr”** ecc.. oppure **“pn”, “ps”** ritorna 1.

Funzione gruppi2

Alla funzione viene passata la variabile **$app** dal programma chiamante e se trova una delle combinazioni **“cn”, “cs”, “pn”, “ps”** ritorna 1.

Funzione inizias

Alla funzione viene passata la variabile **$app** dal programma chiamante e se inizia per **“s”** ritorna 1.

Funzione spezza

Alla funzione viene passata la variabile **$app** dal programma chiamante, il tipo di grafema se consonantico o vocalico e le coordinate per l'inserimento del carattere separatore, per dividere eventuali iati o nessi consonantici non consentiti.

Funzione uguali

Alla funzione viene passata la variabile **$app** dal programma chiamante e se i due caratteri sono uguali o se sono **“c”** e **“q”** ritorna 1.

2.3.4 LA CARTELLA FUNZIONI SPAGNOLO

La cartella contiene le funzioni richiamate dal convertitore spagnolo. Alcune funzioni di sillabazione pur risultando uguali per le due lingue possono dare risultati esattamente opposti: per

esempio in italiano, tutte le sillabe inizianti per **"s"** formano gruppo a sé mentre in spagnolo la **"s"** deve essere sempre scissa dalle altre. La differenza perciò, non risiede nella funzione stessa, ma nella sua interpretazione da parte del programma chiamante.

2.4 LA CARTELLA CORREZIONE

All'interno della cartella correzione sono presenti le quattro pagine che permettono la selezione e l'aggiornamento delle proprietà delle parole che presentano errori nella trascrizione.

Analogamente al programma principale, troveremo nelle intestazioni delle varie pagine, le dichiarazioni delle variabili di sessione o delle istruzioni di inclusione per le funzioni necessarie alle elaborazioni richieste. In ognuna delle pagine troveremo le variabile di sessione relative all'autenticazione, permettendone l'utilizzo solo agli utenti autorizzati.

La prima pagina di correzione visualizza semplicemente all'interno della tabella tutte le parole che potrebbero subire un'eventuale modifica di una proprietà, escludendo quelle che non possono in nessun caso presentare errore. Per un bisillabo non ci sarà mai la possibilità di ambiguità nella accentazione ma al massimo, nella associazione fra grafemi e fonemi: la parola **pala** non dovrà mai subire un'eventuale correzione a differenza della parola **polo**, che potrebbe presentare un errore nella pronuncia della **"o"**, oppure la parola casa che potrebbe averlo per la **"s"**. Al solito, i controlli sui grafemi contenuti in una parola, sono effettuati con le espressioni regolari. Le parole da modificare sono selezionate tramite caselle di spunta di tipo **check box** e vengono inviate alla seconda pagina di correzione. In questa viene creato uno schema in cui sono riportate le parole e tutte le caratteristiche potenzialmente modificabili. Tramite codice **Javascript**, è stata realizzata una procedura di esclusione delle selezioni che non possono essere contemporaneamente applicate ad una parola come già descritto nell'interfaccia grafica. Questa funzione, chiamata **stacca**, racchiude in un gruppo le proprietà **"accento", "iato"** e **"dittongo",** ed in un altro quello della **"o"** e della **"e"** aperta cosicché, cliccando su una di queste proprietà, vengono escluse quelle dell'altro gruppo. Il funzionamento è abbastanza semplice: all'interno del corpo della funzione troviamo l'istruzione che disabilita l'utilizzo della singola **check box,** tramite un costrutto di tipo **if else,** comandato da due parametri, **"a"** e **"b",** associati a ciascun gruppo. Nel momento in cui un parametro viene settato ad uno, tutte le **check box** associate a quel parametro vengono disabilitate. Va sottolineato un aspetto molto importante: poiché ogni **check box** è relativa ad una parola diversa, occorre inserire un contatore all'interno della funzione **stacca** che tenga memoria della riga che l'abbia invocata altrimenti disabiliteremmo con un solo clic tutte le **check box** presenti nella tabella. Inoltre, c'è una proprietà intrinseca associata a tutte due gruppi che è quella dell'omografia, infatti due parole potrebbero semplicemente differire per l'apertura o la chiusura della **"e"** o della **"o",** o per la presenza dell'accento su una sillaba anziché un'altra o ancora, per la presenza di uno iato o di un dittongo dello stesso gruppo di vocali. Si pensi alle coppie *vènti-vénti*, *bòtte-bótte*, *principi-princìpi* e *viola-vìola*. Avendo già a disposizione la funzione prima citata, basterà abilitare una **check box** che permetta la selezione della proprietà di omografia per ogni riga in cui la funzione viene invocata. Spuntando questa casella, le pagine successive avranno la segnalazione che la parola in esame, deve essere considerata un omografo ed inserita nell'opportuna tabella. Il funzionamento della pagina è relativamente semplice: ogni casella di tipo **check box** presenta nel suo campo **name** (ricordiamo che insieme al campo **value**, è quello che permette l'invio dei dati da una pagina ad un'altra), un'etichetta di segnalazione del tipo *attiva_x* seguita dall'istruzione **PHP echo $stringa**, che inserisce la relativa parola nel protocollo **HTTP** e la rende disponibile nella pagina successiva. La *x* nell'etichetta può assumere i seguenti valori: **a** per l'accento, **i** per lo iato, **d** per il dittongo, **s** per il tipo s, **z** per il tipo z, **e** per il tipo e, **o** per il tipo o e **m** per gli omografi. Per ogni casella abilitata sarà inviata alla pagina successiva un'etichetta con la relativa parola.

Nella terza pagina di correzione, sempre tramite una tabella, si sceglie la tipologia della proprietà da modificare ed eventualmente la posizione dove ciò deve avvenire: in una parola come **monologo**, in cui vi sono tre sillabe contenenti una **"o"** eventualmente portatrici di accento, occorre poter scegliere la sillaba interessata ed il tipo di o in essa contenuto. Questo principio, va esteso a tutte le caratteristiche presenti nella tabella. Il funzionamento di questa pagina è leggermente più complesso rispetto a quelle viste precedentemente, poiché occorre riepilogare tutte le caratteristiche da modificare per una singola parola effettuando dei controlli multipli sulla stessa. Sia in questa pagina che nella quarta, sono utilizzate due variabili di sessione, una contatore, di cui spiegheremo la funzione nella quarta pagina, ed una detta **$omoatt**, che segnala semplicemente si sia stata attivata la modalità omografi.

La prima operazione effettuata è la creazione di un vettore associativo che contenga nel campo chiave la parola selezionata e nel campo dati tutte le etichette relative alle caratteristiche da modificare. Ciò è realizzato separando le sottostringhe contenute nel dato inviato dalla pagina precedente ed utilizzandone una come indice del vettore e l'altra come informazione. In questo contesto viene anche analizzata l'etichetta *attiva_m* per settare la variabile **$omoatt**. Il ciclo seguente serve esclusivamente per ordinare i dati all'interno del vettore per poterli incolonnare correttamente all'interno della tabella da creare.

Per semplificare l'elaborazione seguente, viene sostituita all'intera etichetta la lettera rappresentante la caratteristica selezionata. Si noti pure che, nell'attributo responsabile del colore della tabella, ci sia un indice, che consente di variare il colore di righe adiacenti, permettendo una migliore lettura delle singole righe. Il vettore creato in precedenza viene scandito tramite due cicli, il primo dei quali seleziona la parola contenuta nel campo chiave ed il secondo tutti i dati associati alla chiave stessa. Se nel campo dati è presente la lettera di attivazione della procedura di correzione, tramite un costrutto di tipo **switch**, l'elaborazione viene indirizzata al blocco relativo, altrimenti non occorre visualizzare nulla nel campo della tabella. All'interno dei vari blocchi componenti lo **switch**, troviamo dei richiami alla funzione **codifica**, in modo da creare la stessa suddivisione con la quale la parola viene elaborata nella funzione **esamina**: la scelta dell'elemento da modificare, scrive nel db le coordinate direttamente nel formato usato in tutte le altre procedure del programma. La parola ottenuta dalla funzione **codifica** viene analizzata elemento per elemento all'interno di ogni blocco attivato ed in base alle caratteristiche riscontrate, si visualizzano nella tabella gli strumenti di immissione dati più adatti. Come accennato in precedenza, se la parola contiene una sola occorrenza della caratteristica da modificare, occorre utilizzare un semplice **input radio** per selezionare il tipo della caratteristica, mentre, se vi fossero più occorrenze della stessa, si rende necessario l'introduzione di un **input** per ogni occorrenza riscontrata. Dato che in questa pagina i dati potrebbero essere inseriti tramite **input** di natura diversa per ogni caratteristica (**checkbox** e **radio**), l'invio dei dati alla pagina successiva viene realizzata usando sempre l'attributo **value**, unificando l'elaborazione a valle. Analogamente alla pagina precedente, ogni dato presenta al suo interno un'etichetta del tipo *ycoord_W* seguita dall'istruzione **PHP echo $stringa** ormai nota per l'associazione delle coordinate scelte ed il loro invio alla pagina successiva. La *y* indica la caratteristica interessata dalla modifica e *W* il tipo della stessa.

La quarta pagina di correzione è la responsabile dell'aggiornamento del database e rappresenta quindi lo stadio più delicato dell'intera procedura di correzione. C'è il rischio infatti, che un aggiornamento casuale della pagina tramite il bottone del browser, provochi un inserimento indesiderato od una cancellazione all'interno del database. Per questo motivo è stata introdotta la variabile di sessione accennate in precedenza **$contatore**, che permette di discriminare da quale punto del programma si sia giunti a questa pagina: questa variabile viene infatti settata dalla terza pagina di correzione e resettata alla fine della quarta. Il controllo inserito all'inizio della quarta pagina verifica il percorso permettendo l'aggiornamento del database nel caso che dati provengano dalla terza altrimenti, visualizza una animazione che mostra le possibili pagine da scegliere. Nell'eventualità ci si accorga di

aver commesso uno sbaglio nell'inserimento dei dati, occorrerà ripetere la procedura di correzione per la singola parola. Analogamente alla terza pagina di correzione, dopo il controllo sulla variabile **$contatore,** troviamo un ciclo che inserisce in un array associativo le parole nel campo chiave e le caratteristiche e le coordinate inviate nel campo dati. Poiché può avvenire che vi siano coordinate multiple per la stessa parola e per la stessa caratteristica come ad esempio nella parola organizzazione, dove esistono due tipi diversi di zeta in posizione diversa, viene effettuato un secondo ciclo che, se trova dei dati multipli associati alla stessa etichetta, li compatta nel formato con il quale è stata realizzata la base di dati. Il ciclo successivo scandisce gli elementi del vettore così riordinato e demanda con il consueto costrutto **switch** l'elaborazione al blocco che tratta la caratteristica riscontrata. In ogni blocco troveremo la medesima struttura di controllo che verifica se la parola immessa debba essere modificata, nel qual caso se come omografo oppure. Tramite la funzione di **accessoadb2** viene effettuata la scrittura nella tabella opportuna. L'inserimento dei dati nel database segue due processi diversi nel caso si tratti di un omografo oppure di una parola singola.

Nel caso di parola singola, per i casi relativi alla **"e","o","s","z"** occorre scrivere nella relativa tabella nel campo **grafema** la parola in esame, nel campo **tipo** la natura del grafema e nel campo **posizione** le coordinate per individuarlo.

Nel caso dello iato occorre eliminare dalla tabella **iato** la parola errata mentre nel caso del dittongo la si inserirà con la relativa coordinata.

Nel caso dell'accento se la parola è presente nella tabella **eccezioni** viene eliminata altrimenti viene inserita.

Nel caso degli omografi, invece, in ogni blocco del costrutto switch occorrerà creare la coppia di parole da inserire all'interno della relativa tabella. Per l'accento questo viene effettuato utilizzando le coordinate indicate nella terza pagina di correzione, sostituendo le vocali interessate con la loro versione accentata. Per le **"e"** e le **"o"** aperte basterà creare due parole che abbiano nelle coordinate indicate i due tipi di vocale. Nel caso di iato e dittongo basterà invece creare due parole che nelle coordinate indicate, presentino una l'unione e l'altra la separazione delle vocali. Da notare che se non fosse stata prevista la modalità omografo, si sarebbe potuto creare un ciclo continuo di correzione inserendo una volta un'istanza dell'omografo ed una volta l'altra. Prima di aggiornare la tabella degli omografi, la parola interessata viene eliminata da tutte le altre tabelle che eventualmente la contenevano.

Da questa pagina, tramite bottoni realizzata con icone grafiche, è possibile ritornare alla pagina di inserimento testo o di presentazione.

2.5 LA CARTELLA SINTETIZZATORE

All'interno della cartella sintetizzatore troviamo il programma principale per la creazione del file di tipo **HL**, la pagina di scelta dei file di tipo **UTF** e le sottocartelle **funzioni_synth**, contenenti tutte le funzioni richiamate dal programma principale e **hl_files** dove saranno salvati tutti i files **HL** creati dal programma.

Nella pagina di scelta **sintesi.php3**, oltre alla formattazione **HTML**, troviamo un'istruzione che legge i files presenti all'interno della directory utilizzata per salvare i dati di tipo **UTF**. Questi files verranno visualizzati all'interno di un menu a tendina. Selezionandone uno, si aprirà una casella di testo nella quale inserire il nome del file di tipo **HL** che sarà conseguentemente creato e salvato nell'apposita directory.

Nel programma principale invece, troviamo un controllo per verificare se sia necessario creare un file di tipo **UTF** ed una per l'eventuale creazione di un file **HL**. Ricordiamo infatti che è possibile creare indipendentemente i due tipi di files l'uno dall'altro. A seconda dell'esito del controllo, viene attivata la procedura di creazione del file **HL**. Come nel caso del trascrittore fonetico, anche questa procedura è stata generalizzata per la creazione di una frase completa. La stringa da sintetizzare contenuta all'interno del vettore **$HTTP_GET_VARS**, viene divisa, tramite la funzione **explode** nelle parole componenti che vengono salvate col medesimo ordine di arrivo all'interno del vettore **$fono**. Il simbolo usato per separare le parole è **"°"**. Il primo ciclo serve a scandire questo vettore, per poter passare ogni parola alle funzioni preposte alla creazione del file, il secondo per inserire una pausa tra una parola e l'altra, ad eccezione dell'ultima.

Prima di descrivere l'elaborazione, occorre sottolineare che anche in questo caso il processo di sillabazione risulta fondamentale per la corretta creazione del file come risulterà evidente nel seguito dell'esposizione. La creazione del file **HL** è realizzato con i seguenti passi :

0) precodifica della stringa con la funzione **prepara stringahlsyn**

1) creazione di un vettore contenente i parametri dell'HLSyn con la funzione **caricavettore**

2) creazione del file di tipo HL con la funzione **crea_hl_file** con i dati del vettore creato in precedenza

2.5.1 SOTTOCARTELLA FUNZIONI SYNTH

Descriviamo nell'ordine con la quale sono richiamate, le tre funzioni presenti.

Funzione preparastringahlsyn

Le prime righe di codice, realizzate con espressioni regolari, servono a sostituire tutte le consonanti geminate con il simbolo della consonante seguita dal simbolo dell'allungamento di durata **":"**. Il controllo seguente serve a verificare se **$fono** è una parola composta da una o più sillabe. Dopodiché esplode la stringa memorizzando ogni sillaba all'interno di un vettore e lo scandisce per trovare la posizione dell'accento. La vocale interessata dall'accento e quindi soggetta alla geminazione vocalica, verrà seguita dal simbolo di allungamento della durata **":"**. Nello stesso ciclo si verifica se vi sono sillabe implicate, nel qual caso si affianca il simbolo di allungamento di durata alla consonante in coda.

Si noti che in questa maniera, le consonanti germinate sono ricondotte al caso di consonanti semplici e considerate appartenenti completamente alla sillaba seguente e non divise fra quella in esame e quella seguente: ciò permette una enorme semplificazione nella creazione del file finale.

Funzione caricavettore

La stringa passata alla funzione, analogamente a prima, viene divisa nelle sillabe componenti e coerentemente con la codifica fatta precedentemente, se viene riscontrato il simbolo di allungamento, si modifica il valore della variabile **$prolunga**, che provvederà ad aumentare la durata temporale della sillaba interessata. Dopodiché viene effettuato un controllo preventivo per stabilire se la sillaba è libera, implicata oppure se contiene clusters. Questa ultima eventualità non è stata implementata per la mancanza dei dati relativi. Vediamo il caso di sillaba libera.

All'interno della variabile **$voc** viene caricato dal database il vettore contenente i valori della vocale della vocale presente nella sillaba. Si verifica poi se vi siano all'inizio della stessa, consonanti semplici, intendendo con queste anche le geminate per quanto detto prima. Se l'esito è positivo, si carica in un vettore di appoggio avente lo stesso indice della sillaba in esame, la riga del database relativo alla consonante e si ricava dal campo **ue** il numero di righe che compongono il fonema. Come controllo aggiuntivo della correttezza dei dati, si è inserito in valore convenzionale **- 1** nella colonna **tempo**, ad indicare che questa riga contiene semplicemente il dato che sarà utilizzato come contatore per caricare tutte le righe componenti il fonema. Dopodiché si esegue il ciclo di caricamento dati all'interno del vettore denominato **$dati**. All'interno del campo chiave troviamo l'indice della sillaba, nel secondo campo l'identificativo della riga del database contenente i dati del fonema relativi ad un istante e, nel terzo campo, i valori dei parametri. Si noti che unendo il simbolo della consonante con l'indice del ciclo di scansione, si ricrea esattamente l'etichetta della riga del database che deve essere caricata, rispettandone la corretta sequenza temporale. Nei campi delle frequenze formanti, saranno caricati i valori delle formanti della vocale sommati ai valori contenuti nella consonante, espressi come offset rispetto ai valore nominali. Con la codifica utilizzata nella tabella, è possibile trattare correttamente tutti gli offsets di ogni vocale utilizzata. L'espressione regolare utilizzata, infatti, individua il valore numerico associato alla vocale che si è caricata e lo rende disponibile per l'operazione suddetta. Alla fine del caricamento, se la variabile **$flagallunga** è asserita, viene aumentata la durata temporale della sillaba. Se invece è asserita **$flagaccento,** viene incrementata la formante **f0** dell'ultima sillaba.

Il caso di sillabe implicate è simile, tranne che in questo caso vi sono due fonemi consonantici da trattare. Tramite l'espressione regolare inserita, si caricano i due fonemi in un vettore e si esegue per ogni iterazione, l'operazione descritta nel caso di sillabe libere. Alla prima iterazione si utilizzeranno tutte le righe del fonema e si leggerà il valore da usare come contatore nel campo **"ue"** , mentre nella seconda, quella relativa alla consonante in coda alla sillaba, si userà il valore nel campo **"ps",** prelevando così, solo i dati significativi del corrispondente fonema. Si noti, che per la codifica fatta nella funzione precedente, la consonante in coda, presenterà automaticamente il simbolo **":"**, così da accedere alla versione geminata del fonema riscontrato. La sequenza viene ripetuta per ciascuna sillaba contenute nella parola. Al termine viene restituito il vettore contenente i dati dei fonemi ordinati nelle rispettive sillabe.

Funzione crea_hl_syn

Questa funzione non fa altro che inserire in un file i valori caricati nel vettore **$dati**, rispettando la struttura del file tipo **HL**. Se l'operazione di apertura del file in scrittura è andata a buon fine, vengono inserite con le prime due istruzioni l'intestazione del file. Dopodiché si scandisce il vettore **$datihlsyn** che conterrà in ogni elemento i valori dei fonemi di una sillaba e presenterà nel suo campo chiave la lettera iniziale: sarà così possibile determinare se la sillaba è costituita da una sola vocale oppure no. Le elaborazioni saranno leggermente differenti perché, nel caso di una sola vocale, non sarà necessario leggere tutti i singoli istanti caratteristici per la sintesi del fonema ma basterà creare due righe per fornire la durata desiderata alla vocale.

Analizziamo il caso del fonema consonantico. Ricordiamo che il primo indice del vettore indica la sillaba considerata, il secondo l'etichetta di una riga relativa ad un istante del fonema, il terzo al parametro della riga selezionata: utilizzando questi tre indici si accede al valore numerico vero e proprio. Detto questo, risulta evidente come le scansioni annidate l'una dentro l'altra servono ad accedere sia al campo contenente il nome del parametro che al suo valore numerico, poiché entrambi devono essere scritti all'interno del file. Una delle difficoltà presenti per realizzare la corretta struttura del file risiede nel dover replicare per tutti i parametri di una riga il valore dell'istante a cui sono associati. Per far questo, il valore del tempo viene memorizzato dentro alla variabile **$time**. A questa viene eventualmente aggiunto il valore di offset per evitare sovrapposizioni fra due sillabe consecutive. Nella variabile **$key3** avremo il nome del parametro da inserire e nella variabile **$value3** il suo valore. Se **$key3** è **fonema** o **tempo** non deve essere inserito all'interno del file. In ogni ciclo si inseriranno i parametri rilevanti, cioè effettivamente presenti nel db e non sostituiti dal simbolo **"*",** con la seguente sequenza:

1) inserimento di **$key3**
2) inserimento di sei spazi bianchi
3) inserimento del valore dell'istante seguito da sei zeri
4) inserimento del carattere tab
5) inserimento del valore del parametro seguito da sei zeri

6) inserimento di un simbolo di carriage return ed uno di new line

Alla fine della scrittura di ogni blocco di parametri, corrispondente ad una interazione del ciclo esterno, occorrerà inserire nuovamente un simbolo di carriage return ed uno di new line. Tramite la variabile **$offset** sarà possibile decidere il tempo intercorrente fra due sillabe consecutive. La funzione restituisce al programma chiamante, il puntatore al file e l'ultimo valore di tempo inserito, così da poter separare all'interno di una frase, due parole consecutive, inserendo un offset a livello di ciclo più esterno, implementato nel programma principale.

Funzione accessohl_fonemi

Questa funzione prende come parametri il nome del fonema e la tabella nella quale eseguire la query: se il dato viene trovato lo si restituisce al programma chiamante, altrimenti ritorna zero.