

Zmartwychwstał Pan Śpiewy Drogi Neokatechumenalnej — Jak skompilować Śpiewnik?

Andrzej Stanisław Odyniec

18 marca 2024

Dokument ten piszę głównie z myślą o sobie, bo chociaż sam kompilowałem Śpiewnik wiele razy, przedtem sam go wyrzeźbiłem no i wreszcie sam napisałem cały zestaw tajemniczych makr do automatyzacji składu, to pamięć ludzka jest zawodna na tyle, że gdy po dziesięciu latach kolejny raz brałem się za pracę nad tym tematem, bardzo wiele rzeczy musiałem sobie z niemalym wysiłkiem przypominać. Jednak gdyby ktoś posługując się poniższymi notatkami skompilował ten dokument albo sam go sobie modyfikował, ucieszyłoby to mnie niezmiernie.

Spis treści

1	Struktura dokumentu	3
1.1	Obsługa akordów	3
1.1.1	Zmiana tonacji zapisu w źródłach	4
1.1.2	Indywidualne tonacje kantora	10
1.1.3	Różne tonacje na warstwach	11
1.1.4	Makrodefinicje <code>upflag</code> oraz <code>upbox</code>	11
1.1.5	Masowe definiowanie akordów <code>chorddef</code> oraz <code>chmajfamdef</code>	12
1.1.6	Definiowanie obrazków <code>chp</code>	13
1.2	Obsługa struktury śpiewnika	14
1.3	Dokument główny	15
2	Rozdziały z pieśniami	16
2.1	Makra do składu nagłówka rozdziału	16
2.2	Makra do składu nagłówka pieśni	16
2.3	Makra do składu tekstu pieśni	21
2.3.1	Przypisy	22
2.4	Personalizacja	23
3	Zarządzanie stronami	27
3.1	Komendy <code>T_EX</code> owe definiujące strony	27
3.2	Skrypt ustalający porządek stron	28
4	Indeks słów — synchronizacja	29
4.1	Metoda generowania listy indeksowej	30
4.2	Budowa słownika zamian	31
4.3	Algorytm synchronizacji	32
4.4	Aktualizacja słowników zamian	35
4.5	Skrypty do generowania indeksu słów	36
4.5.1	<code>words.bat</code>	37
4.5.2	<code>songs.bat</code>	38
4.5.3	<code>glsid.bat</code>	39
4.6	Generowanie i sortowanie indeksu słów	39
5	Warsztat oraz przebiegi kompilacji	40
5.1	Platforma <code>MiK_TE_X</code>	40
5.2	Sortowanie indeksów	40
5.3	Budowa nutek	45
5.4	Budowa całości oraz drukowanie	46
5.4.1	Marginesy na oprawę	48
5.5	Edycyjne środowisko zintegrowane	49

1 Struktura dokumentu

Początkiem jest plik `SP.tex`, który ma klasyczną strukturę L^AT_EXową. Należy pamiętać, że dokument jest mocno spasowany i wszelkie zmiany totalne (w prologu dokumentu) mogą mieć tragiczny wpływ na wiele stron. Każda sekcja w prologu powinna być opatrzona krótkim komentarzem, np.:

```
% pakiety do fontów i~matematyki
% (akordy składane są w~trybie matematycznym)
\usepackage{fontspec}
\usepackage{polyglossia}
\setmainlanguage{polish}
\usepackage{unicode-math}
```

Powyższa sekcja jest związana z przejściem na unikody tak na wejściu jak i w produkowanym PDFie, co wymaga odpowiednich mechanizmów tak przy selekcji fontów jak i przy obsłudze matematyki. Dalej znajdziemy obsługę fontów, grafiki, identyfikacji tak wersji jak i użytkownika (w wypadku indywidualnych transpozycji niektórych pieśni), ustawień w produkowanym PDFie (jak np. strona startowa) aż do:

```
% pakiety do obsługi akordów oraz struktury śpiewnika
\usepackage{chords,spiewnik}
```

czyli włączenia obsługi systemu składu akordów (`chords.sty`) oraz struktury śpiewnika (`spiewnik.sty`).

1.1 Obsługa akordów

Obsługa akordów jest zdefiniowana w stylu `chords.sty`, który można wywołać z jednym z następujących parametrów definiujących proporcje długości do szerokości w ewentualnych piktogramach akordów:

- short** piktogramy akordów będą składane w postaci krótkiej, po dwie odległości międzystrunowe pomiędzy progami, czyli dla układów trójpozycyjnych w proporcji 6×5 a dla układów czteropozycyjnych w proporcji 8×5 ;
- medium (domyślne)** piktogramy akordów będą nieco dłuższe, po trzy odległości międzystrunowe pomiędzy progami, czyli dla układów trójpozycyjnych w proporcji 9×5 a dla układów czteropozycyjnych w proporcji 12×5 ;
- long** piktogramy akordów będą składane w proporcjach wymiarowych jak na gryfie gitary, po cztery odległości międzystrunowe pomiędzy progami, czyli dla układów trójpozycyjnych w proporcji 12×5 a dla układów czteropozycyjnych w proporcji 16×5 .

Składanie symboli akordów nad tekstem opiera się na przywoływaniu akordu komendą zawierającą w nazwie:

1. literę akordu (wielkie litery [FCGDAEH] oznaczają akordy durowe a małe [fcgdaeh] — akordy molowe),
2. podwyższenie [is] albo obniżenie [es, s] (obsługiwane są też wyjątki, jak [es, as oraz b=hes]),
3. rodzaj akordu i jego alteracje z ewentualnymi wariantami (np. [S] oznacza akord dominantowy septymowy).

Literowy opis tonacji z numeru kwinty generuje makro `\cn`. Jedyne parametry musi być liczbą całkowitą w zakresie 1 ... 35 lub -1 ... -35 tak, że tonacja C-dur odpowiada liczbie 16 a tonacja c-moll liczbie -16. Kolejne liczby odpowiadają kolejnym tonacjom w kole kwintowym, przy czym kolejność określa moduł (wartość bezwzględna) argumentu. Tak więc wartości bezwzględne poniżej 16 odpowiadają tonacjom bemolowym a powyżej 16 — krzyżykowym. Na przykład -14 oznacza g-moll a +18 D-dur.

Komendy stawiające akordy przywołują akord wg numeru kwinty na kole kwintowym. Ten numer będzie przed wygenerowaniem literowego opisu tonu ewentualnie zmodyfikowany poprzez dodanie licznika `\transposeQ` tak więc lokalne modyfikacje tego licznika dokonują automatycznej transpozycji tonacji na kole kwintowym, gdyż wszystkie akordy będą tworzone wg przesuniętych numerów kwint na kole.

1.1.1 Zmiana tonacji zapisu w źródłach

Szereg mechanizmów zmieniających tonację w składzie w wersji personalnej kantora (patrz plik `tps.tex`, str. 23) albo składających alternatywne tonacje na warstwach (patrz plik `ocg.tex`, str. 11) nie zmieniają tonacji zapisu, na którą będą wskazywały elementy sygnatury tonacji. Jednak ich wyniki odnoszą się do tonacji zapisu.

Niemniej może się okazać potrzebnym zmiana tonacji w źródłach, co sprowadza się do wyedytowania wszystkich komend akordowych i zastąpienie ich komendami w innej tonacji. Ta czynność jest z natury błędogenna i czasochłonna. Dlatego zrobiłem skrypt, który automatycznie zmienia tonację w źródłach. Skrypt nazywa się `transposition.awk` i ma następującą treść:

```
BEGIN{
#
# Nazwy akordów oraz ich indeksy na kolekwintowym; ujemne dla molowych
#
  chordno["Feses"]=1; chord[1]="Feses";
  chordno["Ceses"]=2; chord[2]="Ceses";
  chordno["Gesese"]=3; chord[3]="Gesese";
  chordno["Desese"]=4; chord[4]="Desese";
  chordno["Asese"]=5; chord[5]="Asese";
  chordno["Esese"]=6; chord[6]="Esese";
```

```

chordno["Heses"]=7; chord[7]="Heses";
chordno["Fes"]=8; chord[8]="Fes";
chordno["Ces"]=9; chord[9]="Ces";
chordno["Ges"]=10; chord[10]="Ges";
chordno["Des"]=11; chord[11]="Des";
chordno["As"]=12; chord[12]="As";
chordno["Es"]=13; chord[13]="Es";
chordno["B"]=14; chord[14]="B";
chordno["F"]=15; chord[15]="F";
chordno["C"]=16; chord[16]="C";
chordno["G"]=17; chord[17]="G";
chordno["D"]=18; chord[18]="D";
chordno["A"]=19; chord[19]="A";
chordno["E"]=20; chord[20]="E";
chordno["H"]=21; chord[21]="H";
chordno["Fis"]=22; chord[22]="Fis";
chordno["Cis"]=23; chord[23]="Cis";
chordno["Gis"]=24; chord[24]="Gis";
chordno["Dis"]=25; chord[25]="Dis";
chordno["Ais"]=26; chord[26]="Ais";
chordno["Eis"]=27; chord[27]="Eis";
chordno["His"]=28; chord[28]="His";
chordno["Fisis"]=29; chord[29]="Fisis";
chordno["Cisis"]=30; chord[30]="Cisis";
chordno["Gisis"]=31; chord[31]="Gisis";
chordno["Disis"]=32; chord[32]="Disis";
chordno["Aisis"]=33; chord[33]="Aisis";
chordno["Eisis"]=34; chord[34]="Eisis";
chordno["Hisis"]=35; chord[35]="Hisis";
chordno["fesese"]=-1; chord[-1]="fesese";
chordno["cesese"]=-2; chord[-2]="cesese";
chordno["gesese"]=-3; chord[-3]="gesese";
chordno["desese"]=-4; chord[-4]="desese";
chordno["ases"]=-5; chord[-5]="ases";
chordno["eses"]=-6; chord[-6]="eses";
chordno["hesese"]=-7; chord[-7]="hesese";
chordno["fes"]=-8; chord[-8]="fes";
chordno["ces"]=-9; chord[-9]="ces";
chordno["ges"]=-10; chord[-10]="ges";
chordno["des"]=-11; chord[-11]="des";
chordno["as"]=-12; chord[-12]="as";
chordno["es"]=-13; chord[-13]="es";
chordno["b"]=-14; chord[-14]="b";
chordno["f"]=-15; chord[-15]="f";
chordno["c"]=-16; chord[-16]="c";
chordno["g"]=-17; chord[-17]="g";
chordno["d"]=-18; chord[-18]="d";
chordno["a"]=-19; chord[-19]="a";
chordno["e"]=-20; chord[-20]="e";
chordno["h"]=-21; chord[-21]="h";
chordno["fis"]=-22; chord[-22]="fis";
chordno["cis"]=-23; chord[-23]="cis";
chordno["gis"]=-24; chord[-24]="gis";
chordno["dis"]=-25; chord[-25]="dis";
chordno["ais"]=-26; chord[-26]="ais";
chordno["eis"]=-27; chord[-27]="eis";
chordno["his"]=-28; chord[-28]="his";
chordno["fisis"]=-29; chord[-29]="fisis";
chordno["cisis"]=-30; chord[-30]="cisis";
chordno["gisis"]=-31; chord[-31]="gisis";
chordno["disis"]=-32; chord[-32]="disis";
chordno["aisis"]=-33; chord[-33]="aisis";
chordno["eisis"]=-34; chord[-34]="eisis";

```

```

chordno["hisiss"]=-35; chord[-35]="hisiss";
i=0;
#
# Sekwencja dopasowań nazw akordów, poczynając od najdłuższych
#
chordname[++]="Feses";
chordname[++]="Ceses";
chordname[++]="Gesese";
chordname[++]="Desese";
chordname[++]="Heses";
chordname[++]="feses";
chordname[++]="ceses";
chordname[++]="geses";
chordname[++]="deses";
chordname[++]="heses";
chordname[++]="Fisis";
chordname[++]="Cisis";
chordname[++]="Gisis";
chordname[++]="Disis";
chordname[++]="Aisis";
chordname[++]="Eisis";
chordname[++]="Hisiss";
chordname[++]="fisis";
chordname[++]="cisis";
chordname[++]="gisis";
chordname[++]="disis";
chordname[++]="aisis";
chordname[++]="eisis";
chordname[++]="hisiss";
chordname[++]="Ases";
chordname[++]="Esese";
chordname[++]="ases";
chordname[++]="eses";
chordname[++]="Fes";
chordname[++]="Ces";
chordname[++]="Ges";
chordname[++]="Des";
chordname[++]="fes";
chordname[++]="ces";
chordname[++]="ges";
chordname[++]="des";
chordname[++]="Fis";
chordname[++]="Cis";
chordname[++]="Gis";
chordname[++]="Dis";
chordname[++]="Ais";
chordname[++]="Eis";
chordname[++]="His";
chordname[++]="fis";
chordname[++]="cis";
chordname[++]="gis";
chordname[++]="dis";
chordname[++]="ais";
chordname[++]="eis";
chordname[++]="his";
chordname[++]="As";
chordname[++]="Es";
chordname[++]="as";
chordname[++]="es";
chordname[++]="B";
chordname[++]="F";
chordname[++]="C";
chordname[++]="G";

```

```

chordname[++i]="D";
chordname[++i]="A";
chordname[++i]="E";
chordname[++i]="H";
chordname[++i]="b";
chordname[++i]="f";
chordname[++i]="c";
chordname[++i]="g";
chordname[++i]="d";
chordname[++i]="a";
chordname[++i]="e";
chordname[++i]="h";
#
# Wczytanie suffiksów dla rodzin komend akordowych
#
while(getline < "chords.sty" > 0){
  if($0~/\\chm.+famdef{[~]}+){
    if($0~/\\chmajfamdef{.}+){
      majorminor=1
    }else{
      majorminor=-1
    }
    split($0,a,"{");
    gsub("}", "", a[2]);
    suffix[a[2]]=majorminor;
  }
}
collect=0;
if(!transQ){
  transQ=1;
  songid="S106MaryjaMatkaKosciola";
}
}
/\\songid/{
  split($0,a,"{");
  currentsongid=a[3];
  gsub("}", "", currentsongid);
  thissong=currentsongid==songid;
  for(ii=1;ii<=collect;ii++){
    print(memory[ii,thissong]);
  }
  collect=0;
}
{ if($0~/\\begin{songhead}/){
  memory[++collect,0]=$0;
  elems=patsplit($0,a,{[~]}*/,/sep);ll="";
  keytone=a[4];gsub("{", "", keytone);gsub("}", "", keytone);
  oldkey=chordno[keytone];
  newkey=oldkey>0?oldkey+transQ:oldkey-transQ;
  a[4]="{"chord[newkey]}";
  for(ii=0;ii<=elems;ii++){ll=ll"sprintf(\"%s\",a[ii]"sep[ii]);}
  memory[collect,1]=ll;
}else if($0~/\\begin{songbody}/){
  print;
  songbody=1;
}else if($0~/\\end{songbody}/){
  print;
  songbody=0;
}else if(thissong&&songbody){
  line=$0;
  i=1;
  do{
    findin=substr(line,i);

```

```

position=index(findin,"\\");
if(!position)break;
i+=position;
found=0;
for(ii=1;ii<=70;ii++){
    n=chordname[ii];
    if(found)break;
    checkin=substr(line,i,length(n));
    if(n!=checkin)continue;
    found=1;
    ch=n;
    mm=chordno[ch];
}
if(!found)continue;
name=checkin;
j=i+length(name);
found=0;
findin=substr(line,j);
match(findin,/^[A-Za-z]+/);
findrest=substr(findin,RSTART,RLENGTH)
if(RSTART){
    for(s in suffix){
        chno=suffix[s];
        if((findrest==s)&&(chno*mm>0)){
            found=1;
            break
        }
    }
}else{
    found=1;
}
if(!found)continue;
pre=substr(line,1,i-1);
tochange=substr(line,i,j-1);
post=substr(line,j);
key=chordno[tochange];
newkey=key>0?key+transQ:key-transQ;
line=pre"chord[newkey]"post;
}while(1);
print(line);
}else if(collect){
    memory[++collect,0]=$0;
    memory[collect,1]=$0;
}else{
    print;
}
}
}

```

W zmiennej `songid` należy podać id pieśni, np. `songid="S222Akeda"` natomiast w zmiennej `transQ` należy podać liczbę kwint, np. `transQ=1`. Skrypt tylko pomiędzy `\begin{songbody}` i `\end{songbody}` wyszuka komendy akordów a jeśli po notacji znajdzie `suffix`, to sprawdzi jego zgodność a następnie podmieni komendę na odpowiednią z tonacji docelowej. Podmieni także sygnaturę tonacji w poprzedzającym `\begin{songhead}`, nawet w kilku, jeśli są zdublowane instrukcjami warunkowymi, np. `\ifnojahwe`.

Uwaga! Tablica `suffix` aktualizuje się automatycznie z `chords.sty` poprzez skanowanie wywołania makr `\chminfamdef` oraz `\chmajfamdef` i aktualny plik `chords.sty` musi być dostępny w tym samym katalogu.

Przykładowe wywołanie:

```
awk -v songid="S222Akeda" -v transQ=1 -f transposition.awk sprekas.tex > sprekas1.tex
```

Uwaga! skrypt ma naturę pomocniczą a nie automatyczną i ma za zadanie jedynie ułatwić przepisanie tonacji podstawowej wybranej pieśni, gdyż jeszcze wiele pieśni ma różne tonacje w śpiewnikach włoskim i hiszpańskim.

Oddzielnie też należy skorygować odpowiednie zapisy w `tps.tex` oraz `ocg.tex`, aby pasowały do nowej tonacji. Do tych czynności można posłużyć się odpowiednio skryptami `transtps.awk` w stosunku do pliku `tps.tex`

```
BEGIN{
  if(!transQ){
    transQ=-1;
    songid="S106MaryjaMatkaKosciola";
  }
}
{ if($0~songid){
  elems=patsplit($0,a,{[~]}*/,sep);ll="";
  keytrans=a[3];gsub("{","",keytrans);gsub("}","",keytrans);
  nums=patsplit(keytrans,b,[^,]*/,bsep);
  ll="";
  for(i=1;i<=nums;i++){
    b[i]-=transQ;
    if(b[i]==0)b[i]="";
    ll=ll"sprintf("%s",b[i]"bsep[i]);
  }
  a[3]="{ll}";
  ll="";
  for(i=0;i<=elems;i++){ll=ll"sprintf("%s",a[i]"sep[i]);
  if(ll!~/\{\}\}\{\/)print(ll);
}else{
  print
}
}
```

oraz `transocg.awk` w stosunku do pliku `ocg.tex`

```
BEGIN{
  if(!transQ){
    transQ=1;
    songid="S106MaryjaMatkaKosciola";
  }
}
{ if($0~songid){
  elems=patsplit($0,a,{[~]}*/,sep);ll="";
  keytrans=a[2];gsub("{","",keytrans);gsub("}","",keytrans);
  nums=patsplit(keytrans,b,[^,]*/,bsep);
  ll="";
  duplicate=0;
  for(i=1;i<=nums;i++){
    b[i]-=transQ;
    if(b[i]==0){
      b[i]=-transQ;
      duplicate=1;
    }
    ll=ll"sprintf("%s",b[i]"bsep[i]);
  }
  if(!duplicate){
    ll=ll"sprintf("'%s",-transQ);
  }
}
```

```

    gsub("}",",",0}",a[3]);
}
a[2]="{"11"}";
11="";
for(i=0;i<elems;i++){11=11"sprintf("%s",a[i]"sep[i])};
if(11!~/\{\}\{\}/)print(11);
}else{
    print
}
}

```

z takimi samymi parametrami wywołania, czyli zmienną `songid` oraz `transQ`.

Warto też sprawdzić sensowność składu akordów w nowej tonacji. W szczególności sprawdzić, czy nie użyto w źródłach komendy `\Transpose` i zweryfikować układ komend `\tone` oraz ewentualnych klauzul warunkowych sprawdzających licznik `\transposeQ`.

1.1.2 Indywidualne tonacje kantora

Jest kilka sposobów, aby zadeklarować zmienione tonacje. Podstawowy, pierwotny sposób polega na zmianie licznika `\transposeQ=n`, gdzie n jest liczbą kwint transpozycji, przy czym liczby ujemne transponują w dół a dodatnie w górę koła kwintowego. Licznik ten jest zerowany na początku każdej kolejnej pieśni przez środowisko `songhead`, więc zmiany będą obowiązywać tylko w bieżącej pieśni. Jeśli będziemy ustawiać licznik wewnątrz środowiska `songhead`, trzeba ustawić go globalnie: `\global\transposeQ=n`, aby po wyjściu ze środowiska `songhead` nadal obowiązywała zmieniona wartość. Metoda ta wymaga ingerencji w kod pieśni, jest trudna do zapanowania nad nią.

Drugą metodą jest użycie komendy `\Transpose` w otoczeniu `songhead`. To makro ma dwa argumenty. W pierwszym podajemy powód zmiany tonacji a w drugim nową wartość licznika kwint, np. `\Transpose{General}{3}`. W pliku głównym śpiewnika ustawiamy powód zmiany tonacji definiując komendę `\def\Reason{General}`. Podczas budowy zostaną wykonane tylko te wywołania komendy `\Transpose`, które opiewają na zdefiniowaną komendę `\Reason`, czyli w tym przypadku tylko te z podanym powodem `General`.

Trzecia metoda jest przeznaczona do współpracy ze skryptem internetowym `transpozycje.html` i polega na wczytaniu pliku `tps.tex`, który zawiera polecenia transpozycji dla wyspecyfikowanych kantorów i pieśni. W pliku tym w efekcie pobrania i skompilowania danych wyprodukowanych przez skrypt znajdują się komendy `\PersonName{id}{nazwa}[opcje] %email=adres` oraz `\Transposition{id}{id-pieśni}{liczba-kwint}{tonacje}`, które po wczytaniu ustawiają transpozycje wymienionych pieśni dla kantora o identyfikatorze `id` wyszczególnionych pieśni w wersji personalnej. Personalizacja jest opisana bardziej szczegółowo w rozdziale „Personalizacja” na stronie 23.

1.1.3 Różne tonacje na warstwach

Pakiet `spiewnik.sty` pozwala zadeklarować wiele transpozycji dla jednej pieśni. Deklaruje się je komendą `\TrAlternate` wskazując pieśń oraz listę liczników kwint (trzeci parametr jest ignorowany). Zazwyczaj komendy te wczytuje się z pliku `ocg.tex`. Wtedy dla każdej z wymienionych tonacji, jej akordy składane nad tekstem zostaną umieszczone na odrębnej warstwie a w nagłówku pieśni zostaną złożone „przyciski” do włączania żądanej warstwy a wyłączenia pozostałych, poprzedzone przyciskiem powrotu do tonacji podstawowej, wynikającej z ustawień licznika `\transposeQ` innymi metodami.

Plik `ocg.tex` pierwotnie został stworzony skryptem z pliku `tps.tex` zawierającego personalne transpozycje wprowadzone przez kilkudziesięciu kantorów, użytkowników skryptu `transpozycje.html`. Zawiera więc najbardziej potrzebne alternatywne tonacje.

Warstwy, chociaż są w specyfikacji PDFa od dekady, nie są obsługiwane przez większość przeglądarek, w tym przez żadną mobilną. Niektóre przeglądarki respektują ustawienie początkowe widoczności warstw ale nie pozwalają na zmianę ich widoczności łącznie. W takiej przeglądarce zobaczymy akordy tylko w podstawowej tonacji. Są też takie przeglądarki, które włączają widoczność wszystkich warstw. Wtedy widać będzie jednocześnie akordy ze wszystkich tonacji, co czyni taką przeglądarkę bezużyteczną do wersji śpiewnika z wieloma tonacjami na warstwach.

1.1.4 Makrodefinicje `upflag` oraz `upbox`

Jedną z metod stawiania akordu jest wysunięcie go nad wiersz z miejsca, w którym użyto komendy, Odpowiedzialne są za to następujące makra:

```
\def\chordskip{1.6ex}  
\def\chordup{1.9ex}  
\def\upflag#1{\raise\chordskip\hbox{\strut}%  
  \smash{\rlap{\raise\chordup\hbox{\red$#1$}}}}  
\def\upbox#1{\raise\chordskip\hbox{\strut}%  
  \smash{\raise\chordup\hbox{\red$#1$}}}
```

Makro `\upbox{}` umieści opis akordu w pudełku o zerowych wymiarach pionowych podniesionym ponad linię bazową o `\chordup{}` i poprzedzając ją wcześniej `\strutem` (czyli pustym znakiem o wymiarach pionowych nawiasu ale o szerokości zerowej) jednak podniesionym do góry na `\chordskip{}`. Tak więc o ile `\chordup{}` określa wysokość stawiania akordów nad linię bazową to `\chordskip{}` dba o odepchnięcie od linii powyżej.

Parametrami tymi można sterować położeniem akordów nad wierszem poniżej (`\chordup`) oraz odstępem akordów od wiersza powyżej (`\chordskip`).

Jednak zmieniając ich domyślne ustawienia należy całą pieśń, wraz z nagłówkiem i treścią zamknąć w grupie (wziąć w dodatkową parę nawiasów klamrowych) aby zmienione definicje dotyczyły tylko tej pieśni a następnych już nie.

Makro `\upflag{}` także umieszcza akord nad wierszem ale dodatkowo w pudełku zerowej szerokości z materiałem wysuniętym w prawo (czyli stawia taką flagę, chorągiewkę z akordem, skierowaną w prawo). Nie zajmuje to przestrzeni pomiędzy literami poza minimalnym zakłóceniem naturalnych podcięć (*kerningu*), które by między literami wystąpiły, gdyby flagi z akordem nie postawiono. Komenda stawiająca akord używa domyślnie makra `\upflag{}`.

Czasami dla utrzymania rytmu pionowego zachodzi potrzeba *odepchnięcia się* od poprzedniego wiersza tak, jakby w bieżącym wierszu był postawiony akord, chociaż akurat go nie ma. Do tego celu służy *jałowy* akord, który jest pusty ale zachowuje pionowe odstępy: `\I` wykorzystująca `\upflag{}`.

1.1.5 Masowe definiowanie akordów `chorddef` oraz `chmajfamdef`

Komendy stawiające akordy robi (raczej wyłącznie wewnątrz `chords.sty`) makro `\chorddef{ }{ }`, definiujące całą rodzinę makr przywołujących opis akordu tak, że pierwszy argument jest liczbą przekazywaną do makra `\cn`, drugi jest przyrostkiem do generowanej makrodefinicji a trzeci — przyrostkiem do generowanego przez tę rodzinę opisu. Np. dla argumentów `{22}{N}{^9}` wygenerowane zostaną makra dla akordów nonowych: `\FisN` — przywołujące „chorągiewkę” nad tekstem, `\tFisN` — przywołujące tekst do włożenia w `\upflag` lub `\upbox`, `\ddFisN` — przywołujące sam opis akordu bez obrazka i `\lyFisN` — przywołujące obrazek akordu o ile jest zdefiniowana komenda rysująca ten obrazek.

Jest wiele zwyczajów opisu akordu. Definiując akordy, ich grupy i rodziny należy wybrać jakąś jedną metodę, w miarę jednoznaczną, szczególnie, gdy chodzi o akordy nietypowe, alterowane.¹

¹Zwykle przyjmuje się, że jeśli nie postawiono indeksu górnego, to akord jest trójdźwiękiem, czyli jego najwyższym stopniem jest kwinta. Jeśli jednak potrzebujemy dalszych stopni w akordzie, to stawiamy je w indeksie górnym tak, że X^7 oznacza akord-czterodźwięk z dodaną septymą w odległościach pomiędzy stopniami takimi jak w przypadku akordu dominantowego. Jeśli dodamy kolejną tercję do akordu, uzyskamy pięciodźwięk — akord nonowy X^9 . Można dodać jeszcze jedną tercję do akordu aby otrzymać sześciodźwięk undecymowy X^{11} . Formalnie więc górny indeks ma specyfikować najwyższy stopień akordu z założeniem, że poniżej tego stopnia są obecne wszystkie tercje. Trudno jest zagrać wszystkie sześć dźwięków, toteż niektóre się pomija, co zaznacza się zwykle w indeksie dolnym ze znakiem minus, np. X_{-}^9 oznacza akord nonowy ale bez septymy co można by także zapisać jako X_{2+} czyli trójdźwięk z dodaną sekundą (nona w przewrocie jest sekundą). Jednak taki zapis nie jest jednoznaczny, gdyż dodanie interwału poniżej najwyższego stopnia akordu może być rozumiane jako zastąpienie innego dźwięku — tutaj mogłoby to być rozumiane jako trójdźwięk z dodaną sekundą zamiast tercji. Jednak zapis formalnie purystyczny często jest tak skomplikowany, że ludzie preferują uproszczenia, np. wolą X^4 zamiast X_{7-9-}^{11} co

Ponieważ zwykle przyrostek do opisu będzie stawiać indeksy górne i dolne, należy komendy stawiające akordy wywoływać w trybie matematyki. W trybie matematyki używa się specjalnych fontów a symbole tak litery podstawy akordu jak i stopni mogą wymagać przywołania fontu tekstowego, dlatego oznaczenie podstawy akordu zamykamy w komendach `\chordsmaj` oraz `\chordsstop` dla akordów durowych i `\chordsmin` oraz `\chordsstop` dla akordów molowych. Indeksy górne i dolne zamykamy natomiast w pary komend `\chordssub` oraz `\chordsstop`. Komendy te mogą być zdefiniowane w razie zabawy z fontami.

Aby nie definiować grupy akordów dla każdego dźwięku na kole kwintowym, ułatwieniem są komendy definiujące iteracyjnie rodziny akordów dla całego koła kwintowego tak dla akordów durowych `\chmajfamdef{}{}{}` oraz molowych `\chminfamdef{}{}{}` otrzymujące tylko dwa ostatnie argumenty komendy `\chorddef{}{}{}`.

W tym trybie pracy definicja wszystkich akordów septymowych dominantowych wyraża się komendą `\chmajfamdef{S}{~\chordssub7\chordsstop}`. Warto tutaj zauważyć że przyrostek komendy może być tylko literą. Cyfry w \TeX u kończą komendę i do niej nie należą. Dlatego nie mogliśmy zrobić dla akordów septymowych komend postaci `\X7` ale raczej `\XS`.

1.1.6 Definiowanie obrazków `chp`

Dość skomplikowane jest makro `\chp{}` generujące obrazek akordu. Wywołanie ma format: `\chp{p1,p2,p3,...,pn}`, gdzie p_i jest ciągiem trzyznakowym postaci: *osp* lub dwuznakowym postaci: *sp*.

Znak *s* oznacza strunę i jest jedną z liter: ‘e’, ‘h’, ‘g’, ‘d’, ‘a’, ‘E’ odpowiadających strunom E1, H2, G3, D4, A5 i E6. Znak *p* jest cyfrą w zakresie 1–9 i oznacza pozycję (pole między progami), na której należy umieścić palec. Pozycja liczona jest względem początku wycinka chwytни (gryfu). Znak *o* oznacza sposób zaznaczenia umieszczenia palca i może być literą: ‘F’ (generuje czarną kropkę) oznaczającą obowiązkowe umieszczenie palca, ‘f’ (generuje okrąg) — oznacza dodatkowe ułożenie palca albo literą ‘u’ (generuje dwie pionowe kreski albo znak „x”) co oznacza tłumienie nieużywanej struny. Brak znaku *o* jest równoważne literze ‘F’.

Pierwszy parametr może na pozycji *s* mieć literę ‘P’, ‘p’, ‘B’ lub ‘b’, wtedy wybiera on sposób oznakowania wycinka chwytни. I tak wielka litera oznacza wycinek czteropolowy a mała litera — trzypolowy. Litera ‘B’ lub ‘b’ oznacza chwyt barrè — wtedy pierwszy próg zostanie poprzedzony dodatkową kreską. W takim parametrze cyfra na pozycji *p* oznacza numer pierwszej pozycji wycinka chwytни. Brak takiego parametru jest równoważny umieszczeniu go w postaci ‘b1’.

miałoby oznaczać trójdźwięk z dodaną kwartą zamiast akordu undecymowego bez septymy i nony. Jeżeli już, to akord ten powinien być oznaczony jako X_{4+} .

Popularny akord a-moll złożymy wywołaniem: `\chp{h1,g2,d2}`. Akord C-dur złożylibyśmy rozkazem: `\chp{h1,d2,a3,fE3}`, akord d-mol rozkazem: `\chp{e1,g2,h3,uE1}` natomiast akord c-moll: `\chp{b4,h1,g2,d2}`. Nie ukrywam, że zmienna liczba parametrów makra `\chp` skomplikowała nieco jego kod i tak trudny. Dodam więc tylko, że obrazek składany jest przy pomocy klasycznego środowiska semigraficznego \LaTeX a o nazwie `picture`.

Pozostaje mieć nadzieję, że po powyższych uwagach kod `chords.sty` będzie w miarę czytelny.

1.2 Obsługa struktury śpiewnika

Ponieważ pieśń, będąc niejako rozdziałem publikacji nie jest jednak typowym rozdziałem, nie numeruje się go, w tytule poza nazwą umieszcza się inne informacje a na dodatek wymaga nieco innych metod indeksowania, do pisania pieśni stworzyłem odrębne środowiska \LaTeX owe. Wszystkie te rzeczy są zdefiniowane w pliku `spiewnik.sty`. Używanie tych makr opiszę dalej. Tutaj tylko wspomnę o zawartości.

W pliku z tym stylem znajdziemy:

- zestaw deklaracji i makr do znakowania kolorem kartek
- oznaczenia wykonawcy fragmentu (K, W, D, P, ...)
- obsługę zawłóści przypisów dolnych w pieśniach
- obsługę nagłówka i treści pieśni
- prawy nawias klamrowy do powtórzeń
- mechanizm deklarowania tonacji i budowania oznacznika tonacji w nagłówku
- mechanizm deklaracji autora i budowania indeksu źródłowego
- mechanizm deklaracji adresu (biblijnego, tematycznego) i budowania indeksu analitycznego
- mechanizm deklarowania alternatywnych tytułów i budowania indeksu alfabetycznego

Styl `spiewnik.sty` można wywołać z następującymi parametrami:

nocolorbar podanie tego parametru wyłączy rysowanie w pozycji dolnej paginy barwnego paska odpowiadającego kolorowi kartek z oryginalnego śpiewnika włosko-polskiego. Zamiast tego kolor kartki będzie wypełniał symbol roku danej pieśni. Kolor tak pokazany jest wprawdzie słabo widoczny ale nadal dostarcza informacji o kolorze kartki. Takie rozwiązanie może być korzystne w wypadku drukowania na drukarce atramentowej w celu oszczędzenia drogich atramentów barwnych.

nocolorcode parametr ten ma znaczenie, o ile podano w wywołaniu pakietu także parametr **nocolorbar**. Wyłącza on całkowicie wypełnianie kolorem kartek nawet wnętrza symbolu roku pieśni. Takie rozwiązanie może

być przydatne w wypadku drukowania śpiewnika na kolorowym papierze dobieranym barwą do grup stron.

`colorsymbol` parametr ten jest przydatny, o ile podano w wywołaniu pakietu także parametry `nocolorbar` oraz `nocolorcode`. Przy numerze strony będzie umieszczony piktogram odpowiadający kodowi kolorów. Może być to przydatne do składu przeznaczonego na urządzenia monochromatyczne (np. wyświetlacze na e-papierze).

1.3 Dokument główny

Dokument główny zawiera przywołanie plików z poszczególnymi rozdziałami:

```
\include{spintro}  
\include{sptables}  
\include{spliturg}  
\include{sppreka}  
\include{spkatech}  
\include{spwybran}  
\include{spextra}  
\include{koledy}
```

Każdy z plików zawiera treść lub pieśni z odpowiedniej grupy:

<code>spintro</code>	Strona tytułowa i materiał wstępny
<code>sptables</code>	Spisy treści i indeksy
<code>spliturg</code>	Pieśni liturgiczne
<code>sppreka</code>	Pieśni etapów prekatechumenatu
<code>spkatech</code>	Pieśni etapów katechumenatu
<code>spwybran</code>	Pieśni etapów Wybrania
<code>spextra</code>	Pieśni wyrzucone ze śpiewnika z Dobrym Pasterzem
<code>koledy</code>	Wybrane kolędy

W dokumencie głównym (w prologu) zdefiniowane są makroinstrukcje do kompilacji warunkowej uzależniającej skład od formatu kartki, od osoby dla której skład jest personifikowany, od tego, czy skład ma być z akordami nad wersetami, od tego, jaką rodziną krojów pisma ma być złożony a nawet od tego, czy mają być aktywne wyrugowania z tekstów pieśni wystąpienia imienia Boga Jahwe (nastąpiła taka moda w okolicach roku 2010) czy nie mają być rugowane (moda ta zaczęła wygasać w 2015). Niektóre z tych definicji są makrami a niektóre tzw. `ifami`, które można przełączać (np. komenda `\nojahwefalse` przywróci imię Jahwe w tekstach przełączając działanie komendy `\ifnojahwe`). Szczegółowe zrozumienie będzie wymagać przeczytania kodu dokumentu głównego.

2 Rozdziały z pieśniami

2.1 Makra do składu nagłówek rozdziału

Każdy rozdział z pieśniami zawiera kilka deklaracji podobnych do poniższych:

```
\advance\count1 by 1
\pgc[precate]{precate}
\nextside
\pagenumbering{arabic}
\setcounter{page}{31}
\refstepcounter{section}
\addcontentsline{toc}{section}{Pieśni Prekatechumenatu}
```

Deklaracje te odpowiedzialne są za ewentualną aktualizację licznika grupy stron, określenie koloru paska (kolory są z góry określone na początku pliku `spiewnik.sty`). Po przejściu do nowej strony określona jest metoda numeracji stron, może być określony numer strony (jeśli zmieniamy kolejność), podbijany jest numer rozdziału (numery podrozdziałów podbijane są automatycznie w nagłówkach pieśni) oraz dodawana jest do spisu treści nazwa rozdziału.

Po tym wstępie tworzony jest opis każdej pieśni (rozdziału).

Interpretacja nagłówków tworzy wpisy w plikach typu `toc`. Spis treści jest w pliku `SP.toc`. Dane alfabetyczne trafiają do pliku `SP.ial` ale są wczytywane z pliku `SP.ils` natomiast dane analityczne do pliku `SP.ian` ale są wczytywane z pliku `SP.ias`. Jeżeli zmiany w plikach Źródłowych wpłynęły na wpisy w plikach `SP.ian` oraz `SP.ial`, należy posortować wiersze z tych plików wpisując wyniki odpowiednio do plików `SP.ias` oraz `SP.ils`. Należy wziąć pod uwagę fakt, że tak jak źródło, wszystkie pliki są w unikodach, dokładniej w kodowaniu UTF-8. Wypisywane pliki `toc` są bez znacznika BOM (ang. *Byte Order Mark*) na początku i wynik sortowania także powinien być bez znacznika BOM.

2.2 Makra do składu nagłówek pieśni

Nagłówek pieśni tworzy środowisko `songhead`. Oto przykład jego wywołania:

```
\begin{songhead}{Kantyk Jozuego}{e}
\Year{1992}
\address{Joz}{24, 2-13}
\author{AV}
\knownas{Śpiew Jozuego}
\knownas{Dalekie to jest od nas}
\knownas{Po tamtej stronie rzeki}
\end{songhead}
```


Środowisko ma trzy parametry obligatoryjne i trzy opcjonalne (podawane w nawiasach kwadratowych):

tytuł pieśni znajdzie się w spisie treści; będzie też złożony krojem tytułowym nad pieśnią,

podtytuł znajdzie się w indeksie alfabetycznym oraz kursywą obok tytułu głównego,

tonacja zapisu trafi na pierwszą pozycję na liście a potem trafi w lewy górny róg pieśni.

Tak **tytuł pieśni** jak i **podtytuł** mogą być poprzedzone parametrami opcjonalnymi w nawiasach kwadratowych. Parametry te podają wersje skrócone tytułu i podtytułu. Wersje skrócone zostaną użyte w indeksach o ile w prologu ustawiono `\shorttitletrue`. Przydają się w indeksach składanych w wąskich łamach, np. w dwułamowym składzie indeksów na kartkach o szerokości 21 cm. Z kolei **tonacja** także może być poprzedzona opcjonalnym parametrem zawierającym oczekiwany numer strony; jeżeli pieśń podczas składu wypadnie na stronie innej, zostanie zgłoszony błąd.

Jeżeli wersja skrócona podtytułu (w nawiasach kwadratowych) zaczyna się od znaku gwiazdki, zostanie on usunięty a w indeksie alfabetycznym pozycja zaczynająca się od podtytułu się nie pojawi (podobnie jak w tytułach obcojęzycznych). Taki podtytuł ma charakter raczej komentarza do pieśni, niż tytułu alternatywnego.

Jeżeli wersja skrócona tytułu (także w nawiasach kwadratowych) zaczyna się od znaku gwiazdki, cały tytuł zostanie pominięty w spisie treści i w indeksach. Ten mechanizm przewidziany jest dla stron „wielokrotnych”, na których jest kilka wersji takiego samego śpiewu, np. kilka wersji antyfon, aklamacji itp. Aby nie zaśmiecać spisu i indeksu pozycjami różniącymi się wyłącznie jakimś numerem wersji, Wszystkie środowiska `{songhead}` poza pierwszym należy oznaczyć gwiazdką w pierwszym parametrze opcjonalnym (tytuł skrócony) znakiem gwiazdki.

Jeżeli wersję skróconą tytułu poprzedzimy znakiem wykrzyknika, tytuł i podtytuł będą trafiać do spisów i indeksów, ale treść tytułu zostanie wzięta z tytułu strony podanego komendą `\pageid{}` podtytuł w spisach i indeksach będzie pusty. Tak należałoby oznaczyć pierwszy śpiew na stronie, aby wersja numerowana tytułu nie trafiała do indeksów.

W tym środowisku nie umieszczamy tekstu a jedynie wywołania komend indeksujących. Komendy te mogą być używane wielokrotnie i jedno wywołanie nie kasuje drugiego. Są to:

- `\tonef{}` — deklaruje tonację dopisując ją na końcu listy tonacji. Pierwsza tonacja podana tą komendą jest obowiązująca i będzie podana na liście tonacji bez nawiasów, pozostałe będą już w nawiasach. Jeżeli pierwsza tonacja jest identyczna z tonacją zapisu, to jest usuwana z listy i wte-

dy na liście nie będzie tonacji bez nawiasów (wszystkie będą opcjonalne, bo obowiązuje tonacja zapisu).

- `\address{ }{ }` — deklaruje adres do indeksowania. Adres składa się ze skrótu oznaczającego księgę w pierwszym argumencie oraz wersetów w drugim. Skróót oznaczający księgę musi brzmieć identycznie jak to jest wyspecyfikowane w `spiewnik.sty`. Wersety należy wprowadzać tak samo we wszystkich podobnych adresach, bo literalnie wg tegoż tekstu dane będą sortowane.

Adres może też określać przynależność grupową (mszalne, liturgiczne itp.). Wtedy jest skrótem trzyliterowym zaczynającym się od gwiazdki a drugi argument podaje się pusty (wyjątkiem są Ody Salomona, gdzie podaje się numer ody).

Zamiast pustego drugiego argumentu można podać w nim znak gwiazdki, np. `\address{*MSZ}{*}`. Wtedy tak wygenerowana pozycja indeksowa nie będzie zawierać podtytułu a jedynie tytuł główny. Jeżeli jednak środowisko `songhead` jest otoczone przez komendy `\pageid` oraz `\endpageid`, to zamiast tytułu głównego zostanie użyty tytuł strony. Może to być przydatne wtedy, gdy indeksujemy tematycznie podobne pozycje umieszczone na jednej stronie (np. doksologie czy aklamacje) i aby nie mnożyć pozycji bliźniaczo podobnych, umieszczamy komendę `\address` tylko w pierwszej a nie chcemy aby w indeksie uwidoczniany był mylący w takiej sytuacji podtytuł typu *Melodia pierwsza*.

Dopuszczalne są wyłącznie skróty wyspecyfikowane w `spiewnik.sty`. Obecnie obowiązują następujące:







Ody Salomona: *ODA, Okresy Liturgiczne — Adwent: *OAD, Boże Narodzenie: *OBN, Wielki Post: *OWP, Okres Wielkanocny: *OPA, Zesłanie Ducha Świętego: *OPE, Śpiewy Liturgiczne — Śpiewy Mszalne: *MSZ, Pieśni na wejście: *WEJ, Znak pokoju/Przygotowanie darów: *PAX, Śpiewy na Łamanie Chleba: *LAM, Komunia: *KOM, Pieśni na zakończenie: *ZAK, Śpiewy Liturgii Pojednania: *POK, Hymny: *HYM, Śpiewy Liturgii Paschalnej: *PAS, Melodie Hebrajskie: *HEB, Pieśni dla dzieci: *DZI, Liturgia Godzin: *BRE, Pieśni Maryjne: *MAR, Kolędy: *KOL, Inne: *INN.

- `\author{ }` — określa autora śpiewu; zwykle chodzi tutaj o autora tekstu. Jest to także skrót, który musi być wyszczególniony w `spiewnik.sty`. Obecnie są to K: Kiko Argüello, EA: Efraim Abileah, GA: Guglielmo Amadei, HB: Hugo Blanco Manzo, GB: Giuliano Bonomi, SB: Stefan Bortkiewicz, GC: Giacomo Calabrese, NC: Nazareno Cometto, SC: Shlomo Carlebach, LD: Lucien Deiss, GF: Giorgio Filippucci, JG: Jan Karol Gall, EG: Eliyahu Gamliel, FXG: Franz Xaver Gruber, GG: Giuseppe Gennarini, JH: Josef Hadar, MJ: Mateusz Jeż, FK: Franciszek Karpiński, KK: Katarzyna Kulisiewicz, KKR: Karol Kurpiński, TK: Teofil Klonowski, JL: Józef Łaś, GDL: Gigi De Lazzaro, TL: Teofil Lenarto-

wicz, JM: Joseph Mohr, PMz: Pino Manzari, PM: Paolo Marciani, ZO: Zygmunt Odelgiewicz, GR: Giorgio Ricci, PR: Paolo Rita, RR: Roberto Rende, LSa: Luigi Sanna, FVS: Félix Villegas Sanz, AS: Andrea Selloni, LS: Leopold Staff, PS: Piotr Skarga, PSt: Piotr Studziński, AU: Antoni Uruski, AV: Antonio Voltaggio, FV: Franco Voltaggio, trm: melodia tradycyjna, trt: tekst tradycyjny, tr: tradycyjna, U: autorzy poszukiwani. Sortowanie odbywa się według skrótów a nie nazwisk, które są podane w definicjach stylu.

- `\knownas [] {}` — specyfikuje tytuł alternatywny, który trafi do indeksu alfabetycznego. Podobnie, jak w `\songid`, parametr może być poprzedzony jego wersją skróconą, używaną w indeksach, gdy włączony jest `\shorttitletrue`.

Poza komendami indeksującymi można użyć tutaj też komend zdefiniowanych w prologu śpiewnika:

- `\Year{}` — pieczętka roku; jest to jedynie aktualizacja; pieczętka ta stawiana jest na pasku numeru każdej strony według ostatniej aktualizacji i będzie obowiązywać aż do następnej aktualizacji;
- `\warning` — znak ostrzegający , że coś z tą pieśnią jest nie tak, zostanie umieszczony w tytule pieśni; znak ten zazwyczaj sygnalizuje, że pieśń nie ma w ostatnim oficjalnym wydaniu śpiewnika włosko-polskiego a figuruje w rękopisie jedynie ze względu na obecność w innych wydaniach tak polskich jak i zagranicznych; można się spodziewać dezaprobaty różnych osób w wypadku próby śpiewania tak oznaczonej pieśni na liturgiach niezależnie od etapu;
- `\temporary` — w tytule pieśni zostanie umieszczony znak , oznaczający, że pieśń nie było w ostatnim oficjalnym wydaniu śpiewnika włosko-polskiego ale była dana na którejś z późniejszych konwencji rozpoczęcia roku. Jest to więc pieśń jak na razie oficjalna, ale nie ma przydzielonego oficjalnie numeru strony i jej położenie nie jest jeszcze stałe; może się zmienić nie tylko co do numeru strony ale także co do działu (koloru kartek).
- `\revised` — w tytule pieśni zostanie umieszczony znak , oznaczający, że pieśń została w śpiewniku hiszpańskim zrewidowana przez Kiko a tutaj jego poprawki zostały uwzględnione.
- `\esrevised` — w tytule pieśni zostanie umieszczony znak , oznaczający, że pieśń została w śpiewniku hiszpańskim zrewidowana i poprawiona ale bez odwoływania się do osobistej ingerencji Kiko.
- `\keychanged` — w tytule pieśni zostanie umieszczony znak , oznaczający, że pieśń zmieniła tonację zapisu akordów, zazwyczaj na niższą, niż zapis włoski czy nawet hiszpański.
- `\alternatepresent` — w tytule pieśni zostanie umieszczony znak , oznaczający, że pieśń ma wersję alternatywną do wersji z wydania oficjal-

nego, pochodzącą np. ze śpiewnika hiszpańskiego albo z tekstem w innym przekładzie lub z poprawionymi grubymi błędami. Komenda ma parametr opcjonalny (podawany w nawiasach kwadratowych) wskazujący identyfikator pieśni w wersji alternatywnej. Wtedy kliknięcie w symbol przenosi pomiędzy wersjami alternatywnymi.

- `\itdifferent` — w tytule pieśni zostanie umieszczony znak ①, oznaczający, że pieśń była zrewidowana ale włoska wersja na tyle istotnie różni się od hiszpańskiej, że korekt nie można było w całości uwzględnić.
- `\children` — w tytule pieśni zostanie umieszczony znak 𐀀𐀁𐀂, oznaczający, że pieśń przeznaczona jest głównie dla dzieci.
- `\yhwh` — w tytule pieśni zostanie umieszczony obraz hebrajski tetragramu יהוה, oznaczający, że tekst pieśni zawiera imię Boga Jahwe; informacja ta jest czasami potrzebna kantorowi, gdy z różnych powodów nie powinniśmy wymawiać (śpiewać) tego Imienia.
- `\resXXspain[]{}{} , \risMMXXitaly[]{}{} —` (nazwa komendy może być zmieniona, jeśli dodatkowo podano tytuły w różnych wydaniach; wtedy definicje makr decydują, który wariant komendy jest aktualnie aktywny) tytuł w śpiewniku hiszpańskim i włoskim oraz strona w tym śpiewniku; zostanie to złożone po prawej stronie pieśni pionowo a w wersji kwadratowej tytuł hiszpański będzie łączem do tej pieśni w osadzonym śpiewniku hiszpańskim (jak długo śpiewnik ten będzie dostępny w formacie PDF); może zawierać opcjonalnie krótką wersję tytułu do umieszczenia w indeksie zamiast wersji pełnej, przy czym jeśli zamiast krótkiej wersji umieścimy znak gwiazdki, tytuł obcojęzyczny w ogóle nie będzie umieszczany w indeksie — przydaje się to wtedy, gdy tytuł jest bliźniaczo podobny do innego, ale nie identyczny albo gdy kilka wariantów śpiewu występuje pod tym samym tytułem.
- `\resXXtone[]{} , \risMMXXtone[]{} —` tonacja w śpiewniku hiszpańskim lub włoskim, podawana zwykle wtedy, jeśli jest inna, niż w polskim; Jako parametr opcjonalny można podać dodatkowo tonację kapodastrową którą zostanie złożona zaraz za tytułem obcojęzycznym.

Zamiast nagłówka pieśni można umieścić komendę:

`\continued`

Jest ona przeznaczona do kontynuacji pieśni na następnej stronie. Pieśni należy kontynuować na rozwarciu (poczynając od strony parzystej), aby można było śpiewać nie przewracając kartek.

Pieśni długie, nie mieszczące się na jednej stronie i kontynuowana na kolejnej stronie nagłówkiem `\continued`, na końcu strony poprzedniej powinny zawierać informację, że pieśń jest kontynuowana na stronie następnej, chociażby dlatego, by zaaferowany kantor nie przerwał pieśni w połowie, myśląc, że to już koniec. Przed zakończeniem strony poprzedniej warto umieścić komendę

`\verte`

która umieści prawym dolnym roku strony tekst kontynuacja na następnej stronie... — tekst ten będzie wysunięty z dolnej paginy strony. Wysunięcie odpowiada `\footskip` plus wysokość pudełka zebranych przypisów plus 2.6pt. Jeśli ten tekst miał kolidować z przypisami albo był zbyt wysoko, można go przesunąć opcjonalnym argumentem liczbowym, który podaje wysokość podniesienia, jako wielokrotność stopnia pisma przypisów dolnych (na jedną linię przypisu przypada wartość 1.2).

2.3 Makra do składu tekstu pieśni

Po nagłówku pieśni (lub po komendzie `\continued`) umieszczamy treść pieśni w środowisku `songbody`. Gdyby tekst nam się nie mieścił, możemy alternatywnie użyć środowiska `songbodye`, które używa nieco mniejszego stopnia pisma (11pt zamiast 12pt) albo `sonbgodyt`, które sięga po pismo w stopniu 10pt.

Wewnątrz tego środowiska obowiązują nieco inne zasady, niż w trybie poziomym składu `TEX`owego, bo jest ono zanurzone w tabeli (dokładniej: środowisko `TEX`owe `\hailgn`), gdzie pierwsza kolumna przeznaczona jest na oznaczenia roli a druga na treść wiersza. Dodatkowo włączony jest mechanizm respektowania znaku końca wiersza, który normalnie jest przez `TEX`a ignorowany.

Wiersz można zacząć jedną z komend oznaczających role (zdefiniowane są one zaraz na początku stylu `spiewnik.sty`). W wierszu można używać komend wstawiających akordy. Należy pamiętać, że każda komenda `TEX`owa „pożera” spacje po niej występujące, chyba że występuje po niej następna komenda albo znak różny od litery (np. pusta grupa `{}`).

Grupę wierszy można wziąć w środowisko `rightbrace` z jednym parametrem. W ten sposób obejmiemy te wiersze klamrą po prawej stronie a pośrodku tej klamry po prawej umieścimy wartość parametru. Środowisko `bis` nie ma parametru ale zdefiniowane jest jak poprzednie lecz z parametrem `2×`.

Wszelki materiał pionowy (np. odstępy pionowe, `\medskip`) można wstawiać jako argumenty komendy `\sep{}`.

Do podziału pieśni na łamy (zwane też kolumnami) można użyć komendy `\splitsong`.

Pudełko pieśni będzie tabelą dwukolumnową. Pierwsza kolumna zawiera oznaczenie roli a druga — treść wersetu. Szerokość tego pudełka jest więc sumą najszerszego oznaczenia roli, najdłuższego wersetu oraz odstępu międzykolumnowego, który ma miarę pół firetu. Pudełko będzie centrowane na stronie klejami. Jeżeli chcemy zredukować wpływ najdłuższego wersetu (gdy jest istotnie dłuższy od pozostałych) tak, aby optycznie przesunąć pudełko w prawo, możemy na końcu takiego wersetu dodać ujemne podcięcie (`\kern`).

Podział na dwa łamy odbywa się poprzez zakończenie tabeli, dodanie poziomego kleju ściskalnego (tym razem `\hss` a nie `\hfil`) i ponownie jej otwarcie. W ten sposób pudełka poszczególnych łamów mogą mieć różne szerokości a kleje zostaną wyrównane, chyba że zostaną zupełnie ściśnięte a klej wewnętrzny może nawet zyskać ujemną szerokość.

Aby nad tym aspektem panować, mamy dodatkowe warianty komendy `\splitsong` oraz środowiska `{songbody}`. Jest więc `\splitsongwp{}{}{}` w której podajemy tekst najdłuższego oznaczenia roli dla drugiego łamu (można użyć gotowych komend, np `\ttKW`) oraz w pozostałych dwóch parametrach minimalne szerokości obu łamów (warto je podawać w wielokrotności stopnia fontu `em`). Uproszczona wersja tej komendy nazywa się `\splitsongw{}{}{}` i nie ma pierwszego argumentu a szerokość kolumny oznaczeń w drugim łamie będzie wynikała wyłącznie z użytych oznaczeń roli. Minimalną szerokość łamu realizujemy poprzez dodanie na spodzie pierwszego oraz szczycie drugiego pudełka poziomej ruli o zerowej wysokości i zadanej szerokości.

Jeśli chcemy panować także nad szerokością kolumny oznaczeń roli w pierwszym łamie lub panować nad położeniem łamów pieśni na stronie, powinniśmy zamiast środowiska `{sobgbody}` użyć środowiska `{songbodyfix}{}{}`, którego pierwszym parametrem jest tekst najdłuższego oznaczenia roli dla pierwszego (lub jedyne) łamu a drugim argumentem jest szerokość pudełka.

Aby łatwiej było manipulować powyższymi parametrami, można włączyć zdefiniowany wewnątrz pakietu `spiewnik.sty` `\iffframes` przy pomocy komendy `\framestrue`. Tak uzyskamy ramki wokół pudełek tekstu pieśni oraz pogrubione rule tam, gdzie wstawiane są jako odstępniki na podstawie szerokości podanych w parametrach środowiska oraz komend dzielących na łamy. Należy pamiętać aby po testach wyłączyć ramki `\framesfalse`.

Tymi metodami możemy usztywnić rozmiary pudełek i szerokości kolumn na tyle, aby dwie pieśni jednołamowe na stronie miały pionowo zsynchronizowany lewy margines tekstu a dwułamowe miały podobnie zsynchronizowane oba łamy. Pozostaje w kwestii wyczucia estetyki decyzja, do jak wielkich rozbieżności w szerokości łamów występujących na sobą należy jeszcze synchronizować lewy margines a od jakich rozbieżności poczynając należy stawiać na centrowanie pudełek występujących nad sobą.

2.3.1 Przypisy

Specjalnego traktowania wymagają przypisy dolne, których poza tabelę nie można wyprowadzić. Dlatego w pieśni nie można użyć komendy `\footnote{}`, bo treść przypisu zginie.

Zamiast tego możemy użyć dwóch komend: wewnątrz tekstu pieśni komendy `\footnotemark` a poza środowiskiem z treścią pieśni — komendy `\footnotetext{}`. Mechanizm nie wymaga nadzoru jak długo w pieśni jest

tylko jeden przypis. Jeśli chcemy mieć ich więcej, musimy zapanować nad dopasowaniem znaczników do tekstów przypisu.

Jeśli przypisów jest kilka, należy zapanować nad licznikiem `footnote`. Jeżeli jednak używamy pakietu `hyperref` (a jest on niezbędny do hiperłączy), powinniśmy jeszcze zapanować nad licznikiem `Hfootnote` oraz nad etykietami, do których będą występować odwołania. Do takich celów służą poniższe komendy. `\footnotemark` przed postawieniem znacznika podbija oba liczniki a następnie z `Hfootnote` i innych informacji tworzy etykietę do hiperłącza. Ta etykieta zostanie przyporządkowana hiperłączu do przypisu zrobionego przez `\footnotetext`.

`\ftmarklast` cofa oba liczniki a następnie robi znacznik. Taki znacznik będzie wskazywał na ten sam przypis, co znacznik poprzedni.

Jeżeli zamierzamy zrobić dwa przypisy lub więcej, potrzebujemy dodatkowo po postawieniu znacznika zapamiętać etykietę, gdyż następny znacznik stworzy etykietę dla kolejnego przypisu zamazując poprzednią. Komenda `\ftmarksave{\nazwa}` postawi znacznik i zapamięta etykietę hiperłącza w makrze `\nazwa`.

Komenda `\fbackrestore{liczba}{\nazwa}` przywoła zapamiętaną etykietę oraz doda do liczników parametr `liczba`. Umożliwia to sterowanie zarówno licznikami decydującymi o wartości znaczników jak i etykietą, która będzie przydzielona przypisowi.

Dzięki sprawnemu manipulowaniu licznikami i etykietami można uzyskać odpowiednie mapowanie pomiędzy znacznikami przypisów oraz ich treściami pozwalając na wspólne przypisy dla wielu znaczników. Zaleca się jednak przetestować działanie hiperłączy oraz dopilnować aby nie było przypisów niezwiązanych ze znacznikami.

Warto zgodnie ze zwyczajami druków liturgicznych komentarze drukować na czerwono. Może do tego służyć komenda `\red` zdefiniowana w `chords.sty`, którą powinno się umieścić w treści przypisu.

2.4 Personalizacja

Każda pieśń zapisana jest w pewnej konkretnej tonacji. Tonacja notacji jest oznaczona w ostatnim parametrze środowiska nagłówkowego `songhead`. Poza tym w każdym nagłówku można podać tonacje, w których pieśń należy wykonywać. Robimy to poleceniem `\tone{}`. Można to zrobić kilka razy. Wtedy pierwsza pozycja będzie składana w nagłówku bez nawiasów (jako obowiązująca) a pozostałe jako opcjonalne. Jeżeli pierwsza z tak podanych tonacji jest identyczna z tonacją zapisu, nie jest ona w nagłówku powtarzana (bo obowiązuje tonacja zapisu) a wszystkie pozostałe będą złożone w nawiasach.

Każda pieśń powinna mieć podany w nagłówku identyfikator pieśni. Robi się to poleceniem postaci:

```
\songid{S}{S102PanPoteznyOglaszaDobraNowine}
```

gdzie drugi parametr jest unikalnym identyfikatorem, używanym dla danej pieśni tak w śpiewniku, jak i w nutach (jest także nazwą pliku z nutami). Pierwszy parametr definiuje lokalizację nut: **L** — liturgiczne, **S** — śpiewnikowe oraz **X** — brak nut. Identyfikator służy do budowania hiperłączy tak wewnątrz tomu jak i pomiędzy tomami.

Identyfikator pieśni umożliwia też odnalezienie jej w zbiorze dla innych potrzeb, np. do personalizacji tonacji.

Aby zmienić tonację pieśni, należy w pliku `tps.tex` w tym samym katalogu, co startowy plik źródłowy śpiewnika tekstowego, umieścić np. komendę:

```
\Transposition{Andrzej-Odyniec}{S219HymnOKrzyżuChwalebny}{-4}{Cis,D}
```

Pierwszym parametrem jest identyfikator kantora. Może on zawierać tylko litery łacińskie oraz kreskę. Musi być identyczny z identyfikatorem składu w nazwie pliku, np. `SP@Andrzej-Odyniec`. Drugim parametrem jest wspomniany identyfikator pieśni, podany w nagłówku. W trzecim jest liczba kwint, o które należy transponować zapis a czwarty zawiera listę dodatkowych tonacji, złożonych w nagłówku. Trzeci albo czwarty parametr może być pusty (nie obydwaj jednocześnie). Makroinstrukcja `\Transposition` ma też dodatkowy, opcjonalny argument wybierający wersję transpozycji dla danego kantora z pozycji komendy `\Personname`.

Podczas składu śpiewnika, do którego nazwy dodano po znaku „@” identyfikator kantora, pieśń wskazana w komendzie `\Transposition{}{}{}{}` będzie transponowana o wskazaną liczbę kwint. Do nagłówka zostaną dodane dodatkowe tonacje. Jeżeli któraś z dodatkowych tonacji została podana jako **x**, zostaną zapomniane wszystkie wcześniej wprowadzone tonacje, tak w źródłach jak i w tej komendzie a tonacja następna po **x** będzie złożona bez nawiasów — jako obowiązująca. Przypomnijmy, że jeśli tonacja pierwsza (ta bez nawiasów) jest identyczna z notacją zapisu (tą w nawiasach trójkątnych), to tonacja bez nawiasów na liście jest pomijana i pozostają wyłącznie tonacje w nawiasach. Obowiązującą jest bowiem tonacja zapisu. Ta zasada obowiązuje niezależnie od tego, czy lista tonacji dodatkowych została podana w źródłach czy w pliku personifikującym.

W tym samym pliku należy raz jeden podać nazwę kantora (już bez ograniczeń, z polskimi literami w kodowaniu UTF8):

```
\PersonName{Andrzej-Odyniec}{Andrzej Odyniec}
```

Na pasku na dole strony, tam gdzie występuje identyfikator wydania (np. Pascha 2018), pojawi się po przeciwnej stronie jako dopisek w nawiasach klamrowych nazwa kantora podana w drugim parametrze.

Makro `\Personname` ma też dodatkowy, opcjonalny argument, podawany na końcu w nawiasach kwadratowych. W argumencie tym podajemy listę opcji ewentualnie oddzielonych przecinkami:

Układ — domyślnie obowiązuje układ klasyczny, gdzie numery stron są powiązane z konkretnymi pieśniami; układ można zmienić opcją:

alpha — skład zostanie zrobiony w układzie alfabetycznym, wprowadzonym do świata włosko-polskiego w roku 2020 wraz z wydaniem śpiewnika włoskiego na wzór hiszpańskiego; w praktyce jest to już nieco inne wydanie, gdyż układ klasyczny nie był od tego czasu pielęgnowany a układ alfabetyczny opiera się na odrębnych plikach źródłowych; patrz też: modyfikator `-alpha` w rozdziale „Budowa całości oraz drukowanie”

Format — domyślnie obowiązuje format A5 bez marginesów; poniższe opcje zmieniają format i kolory:

narrow — skład A5 zostanie zrobiony z poszerzonym na oprawę lewym marginesem i nieco zawężoną kolumną; przy stosowaniu fontów *latin modern* zostaną wykorzystane nieco węższe wersje fontów projektowane dla wielkości optycznej 17pt; patrz też: przyrostek **narrow** w rozdziale „Budowa całości oraz drukowanie”

kikosq — skład w formacie 21 cm×21 cm (w starym układzie klasycznym 21 cm×20 cm) a więc identycznym z wydaniem oficjalnym dla kantora; patrz też: przyrostek **kikosq** w rozdziale „Budowa całości oraz drukowanie”

tab34 — skład w formacie 15,75 cm×21 cm, wypełniającym ekran tabletu o proporcjach boków 3×4; patrz też: przyrostek **tab34** w rozdziale „Budowa całości oraz drukowanie”

mono — skład monochromatyczny, do druku na drukarce czarno-białej; patrz też: modyfikator `-mono` w rozdziale „Budowa całości oraz drukowanie”

Font — domyślnie skład robiony jest rodziną krojów Latin Modern, od zawsze związaną z \TeX em; opcją można to zmienić:

minion — skład ma być zrobiony nie rodziną fontów *latin modern* ale *minion/myriad/chapparral*; fonty te należy mieć zainstalowane w systemie; patrz też: modyfikator `-minion` w rozdziale „Budowa całości oraz drukowanie”

Akordy — domyślnie akordy składane są jakąś pisanką powiązaną z wybraną rodziną fontów; można to zmienić opcją:

serifchords — akordy zostaną złożone zwykłym pismem szeryfowym a nie pisaną; patrz też: modyfikator **-serifchords** w rozdziale „Budowa całości oraz drukowanie”

Tonacje — domyślnie każda pieśń ma nad tekstem akordy w wybranej tonacji; można użyć poniższej opcji:

layers — akordy zostaną złożone w wielu tonacjach jednocześnie na warstwach OCG; nad tytułem pojawią się przyciski do przełączania warstw (warstwy obsługują obecnie tylko pełne wersje Adobe Acrobat/Reader)

Kod rozdziału — domyślnie rozdziały oznaczane są paskiem barwnym związanym z kolorem kartek wydania oficjalnego; jedną z poniższych opcji można to zmienić:

colorstamp — zamiast kolorowego paska pod paginą dolną pokolorowane zostanie tylko wnętrze pieczęci roku pośrodku dolnej paginy

colorpaper — dolna pagina będzie pozbawiona kodu barwnego — skład będzie drukowany na barwnym papierze

colorsymbol — zamiast kół kolorowych obok numeru strony pojawi się odpowiedni symbol powiązany z rozdziałem

Wariant transpozycji — zadane transpozycje osobiste przewidują jedną tonację dla każdej pieśni; tutaj można opcją wybierać jedną z kilku transpozycji dla tej samej pieśni; opcje są nieobsługiwane przez skrypt **transpozycje.html**:

versiona — z transpozycji zostaną wybrane tylko te, które mają na końcu opcjonalny parametr **[versiona]**

versionb — z transpozycji zostaną wybrane tylko te, które mają na końcu opcjonalny parametr **[versionb]**

versionc — z transpozycji zostaną wybrane tylko te, które mają na końcu opcjonalny parametr **[versionc]**

Jeżeli skompilujemy **SP.tex** z opcją **--jobname=SP@Andrzej-Odyniec**, powyższa definicja dojdzie warunkowo do skutku a wynik znajdzie się w pliku **SP@Andrzej-Odyniec.pdf**. Ponieważ wszystkie pliki pośrednie będą także miały zmienione nazwy, należy do sortowania indeksów użyć innego skryptu: **sortix-personal.ps1** wywołanego z parametrem **Andrzej-Odyniec**.

Do wykonania kompletnej trzyprzebiegowej kompilacji z sortowaniem indeksów należy użyć skryptu **compile3pass-personal.ps1** z podobnie podanym parametrem a do kompilacji broszur ze zrobionego już spersonalizowanego pliku śpiewnika należy użyć skryptu **compile-broszury-personal.ps1** z podobnie podanym parametrem.

Skrypt do deklaracji własnych transpozycji Pod adresem <http://spiewnik.andrzej.odyniec.info/html/transpozycje.html> jest skrypt ułatwiający przygotowanie zestawu komend transpozycji dla kantora. Można taki zestaw utworzyć lub modyfikować. Nie trzeba umieć liczyć kwint. Wystarczy wskazać, jaki akord przechodzi na jaki a liczba kwint zostanie wyliczona przez program. Można też wybrać tonacje do kapodastra. Zestaw transpozycji można zapisać na serwerze. Wtedy zostanie on użyty przy kolejnej budowie a wynikowy plik zostanie umieszczony na serwerze. Można też pobrać plik na swój komputer a potem wykorzystać go podczas własnej kompilacji ze źródeł albo dostarczając go komuś, kto potrafi zrobić taką kompilację.

3 Zarządzanie stronami

Do poprawnego składu zarządzanie stronami jest właściwie niepotrzebne. Użycie odpowiedniej komendy emitującej T_EXowy `\eject` gwarantuje przejście do strony następnej a liczniki zajmą się numerowaniem stron.

Kiedy jednak składa się z tego samego źródła wiele wersji, a treść nie może łamać się na strony dowolnie, bowiem pieśń powinna być w całości na tej samej stronie, potrzebna jest weryfikacja, czy po zmianach nie poszło coś źle i strony nie połamały się niewłaściwie. Do tego celu w opisane wcześniej mechanizmy zostały wpisane odpowiednie zabezpieczenia, jak np. możliwość podania oczekiwanego numeru strony w parametrze środowiska **songhead** (patrz rozdział 2.2 „Makra do składu nagłówka pieśni” na stronie 16).

3.1 Komendy T_EXowe definiujące strony

Ewentualna zmiana kolejności stron byłaby dość trudna. Aby ją ułatwić a potem nad nią zapanować, wprowadzone zostały dodatkowe dwie komendy:

`\pageid{ }{ }{ }` z trzema parametrami obligatoryjnymi i jednym opcjonalnym na końcu:

nazwa strony do sortowania w kolejności alfabetycznej — zwykle jest to tytuł pieśni; należy zauważyć, że na jednej stronie może być kilka pieśni a jedna pieśń może zajmować kilka stron, dlatego przy sortowaniu alfabetycznym do klucza sortowania należy dodać numer strony,

numer strony zamierzonej na której materiał powinien wystąpić,
numer strony adresowej do sortowania klasycznego, który to numer został kiedyś pieśni przypisany i według którego następują odwołania z tomów z nutami,

identyfikator koloru parametr opcjonalny; jeśli kolor jest inny, niż w całym rozdziale;

`\endpageid{}` z jednym parametrem obligatoryjnym

numer strony zamierzonej na której materiał powinien wystąpić, powinien być identyczny z tym w komendzie `\pageid{}{}{}[]`.

Komendy te pozwalają:

- zapisać rzeczywiste położenia stron w pliku `.pid`, który to plik umożliwi w odwołaniach z nut zastąpić klasyczny numer strony z odwołania, numerem rzeczywistym,
- ewentualnie sprawdzić, czy strona zamierzona zgadza się z rzeczywistą,
- ustawić inny kolor dla danej strony, niż kolor rozdziału (np. dla pieśni na łamanie chleba),
- zweryfikować, czy niektóre parametry składu, jak podniesienie od dołu czy odepchnięcie od góry akordu albo zagęszczenie wierszy, nie zostały w sposób niezamierzony zmienione pomiędzy stronami (co może rzutować na nieprawidłowości w składzie)

3.2 Skrypt ustalający porządek stron

Skrypt awkowy `sortsong.awk` przegląda tekst źródłowy rozdziału pod kątem komend określających początek i koniec strony. Skrypt ten jest sterowany wartością zmiennej awkowej `order`, która może mieć wartość `classic` (sortowanie wg numeru klasycznego), `seq` (sortowanie wg zamierzonego numeru kolejnego) albo dowolną inną (sortowanie wg alfabety). Skrypt można wywołać na przykład tak:

```
awk -v order "" -f sortsong.awk sppreka.tex > spprekasort.tex
```

a następnie utworzonym plikiem zastąpić oryginalny. Należy pamiętać o zrobieniu kopii oryginału, gdyby coś nie poszło zgodnie z naszymi oczekiwaniami.

Skrypt wylapuje strony, zapisuje je w awkowej tablicy asocjacyjnej, wypisuje indeks stron do pliku tekstowego, sortuje go skryptem zewnętrznym w powershellu `sort.ps1` a następnie po wczytaniu według niego wypisuje strony w nowej kolejności.

Treść skryptu znajduje się w pliku `sortsong.awk` ale można też zapoznać się z jego treścią poniżej:

```
BEGIN{pageid="intro";
  line[pageid]=0;}
/\pageid/{
  if(pageid!="intro"){
    for(i=0;i<line[pageid];i++){
      lines[lastpageid][line[lastpageid]++]=lines[pageid][i];
    }
  }
  split($0,a,"");
  split(a[1],b,"");
  split(a[3],c,"");
  pageid=c[1];
  pgix[pgi++]=pageid;
```

```

sortkey[pageid]=b[2];
pageseq[pageid]=a[2];
if(substr(c[2],1,1)=="["){
    color=c[2];
    gsub("\\[", "", color);
    gsub("\\]", "", color);
    pagecolor[pageid]=color;
}
line[pageid]=0;
lines[pageid][line[pageid]++]=0;
}
/\endpageid/{
    split($0,a,"{");
    split(a[2],b,"}");
    pagesq=b[1];
if(pageseq!=pageseq[pageid])
    print "Error! inconsistent seq no. ending >>"sortkey[pageid]"<< "pageseq[pageid]"<->"pagesq >
    "/dev/stderr"
    lines[pageid][line[pageid]++]=0;
    lastpageid=pageid;
    pageid=0;
    line[pageid]=0;
}
{ if($0!~/\endpageid/){lines[pageid][line[pageid]++]=0;
}
}
END{
    for(j=0;j<pgi;j++){
        if(order=="seq"){
            print sprintf("%03d",pageseq[pgix[j]])+"sortkey[pgix[j]]"+"pgix[j]"+" > "sortkey.txt";
        } else if(order=="classic"){
            print sprintf("%03d",pgix[j])+"sortkey[pgix[j]]"+"pageseq[pgix[j]]"+" > "sortkey.txt";
        } else {
            print sortkey[pgix[j]]+"pageseq[pgix[j]]"+"pgix[j]"+" > "sortkey.txt";
        }
    }
    system("del sortkey.txt");
    system("start /wait powershell -ExecutionPolicy Unrestricted -Command \"\\.\sort.ps1 sortkey\\\"");
    while(getline < "sortkey.txt"){ print > "/dev/stderr"
        split($0,a,"+");
        if(order=="seq"){
            if(length(a[1])>0)npix[npgi++]=a[3]*1;
        } else if(order=="classic"){
            if(length(a[2])>0)npix[npgi++]=a[1]*1;
        } else {
            if(length(a[2])>0)npix[npgi++]=a[2]*1;
        }
    }
    for(i=0;i<line["intro"];i++){
        print lines["intro"][i];
    }
    for(j=0;j<npgi;j++){
        for(i=0;i<line[npix[j]];i++){
            print lines[npix[j]][i];
        }
        for(i=0;i<line[0];i++){
            print lines[0][i];
        }
    }
}

```

4 Indeks słów — synchronizacja

Indeks słów występujących w tekstach piosenek a właściwie ich form podstawowych jest dość prosty do zapisania ale bez wspomagania programowego dość pracochłonny, gdyż wymaga, aby wszystkie hasła, które mają znaleźć się w indeksie, były wyszczególnione przed tekstem każdej piosenki zaraz po komendzie `\end{songhead}` w postaci wywołań `\index{hasło}`. Synchronizacja tych wywołań z tekstem piosenki albo będzie wspomagana narzędziami

albo stanie się koszmarnie pracochłonna i błędogenna.

4.1 Metoda generowania listy indeksowej

Aby zsynchronizować listę komend indeksujących z treścią pieśni, należy:

1. Wyłuskać z treści wszystkich pieśni słowa; to wymaga „oczyszczenia” treści z komend formatujących i wstawiających akordy, podzielenia wierszy na słowa oraz wyprowadzenie słów w jednej kolumnie do pliku tekstowego a następnie posortowanie tego pliku oraz usunięcie zeń powtarzających się wierszy,
2. Tak uzyskaną listę słów będzie trzeba sprowadzić z form fleksyjnych do form podstawowych; do tego należy stworzyć słownik zamian w którego pierwszej kolumnie będziemy mieć słowo w wersji fleksyjnej a w drugiej w formie podstawowej, np. **dnie dzień**. Jeśli słowo ma być zignorowane w formie fleksyjnej, w drugiej kolumnie powinien widnieć jedynie znak „-”. W taki sposób należy potraktować nieindeksowane słowa obcojęzyczne oraz sekwencje znaków, których algorytmowi oczyszczania tekstu z informacji formatujących nie udało się oczyścić. Jest to raczej naturalne, gdyż robimy to wyrażeniami regułowymi a nie głęboką analizą składniową języka \TeX . Słownik zamian znajduje się w pliku **glsrepl.txt**

Wszystkich słów w formach fleksyjnych jest obecnie prawie 8000 a w formach podstawowych ponad dwukrotnie mniej. Budowa takiego słownika ręcznie jest bardzo pracochłonna, chociaż można ją wspomagać programowo, o czym za chwilę.

3. Może się zdarzyć, że zamiany w słowniku zamian nie będą jednoznaczne, np. **dobra dobra** (Sekwencja na Boże Ciało) **dobra dobro** (Psalm 16: Ty mi ukazesz ścieżkę życia), **dobra dobry** (Weź mnie do nieba). W takiej sytuacji nie można polegać na jednym wzorcu zamiany i należy wpisać wzorec niejednoznaczny, np. **dobra dobra+dobro+dobry**. Takich wpisów jest mało, niewiele ponad 30.

Gdyby na tym pozostać, w indeksie mielibyśmy hasło wieloczłonowe. Aby do tego nie dopuścić, należy już w fazie tworzenia lokalnych wpisów indeksowych dla każdej pieśni posłużyć się słownikiem zamian lokalnych, w którym dla każdej pieśni będzie oddzielny wpis, np.

dobra+dobro+dobry dobro	S135TyMiUkazeszSciezkeZycia
dobra+dobro+dobry dobra	S155ChwalSyjnie
dobra+dobro+dobry dobry	S280WezMnieDoNieba

przy czym pieśni identyfikujemy tutaj identyfikatorami. Słownik zamian lokalnych znajduje się w pliku `glsreplpage.txt`. Może się zdarzyć, że dla danego słowa jest w słowniku zamian globalnych kilka form podstawowych ale w jakiejś pieśni występuje nie jedna ale więcej takich form, bo słowo jest użyte w kilku znaczeniach. Wtedy w drugiej kolumnie należy umieścić wszystkie takie formy i oddzielić je znakiem „&”.

`może+móc` `może&móc S089SpiewBalaama`

Wpisów w słowniku zamian lokalnych jest nieco ponad 300.

O ile budowanie słownika zamian można wspomagać programowo posługując się dostępnym publicznie słownikiem odmian form podstawowych słów, to niejednoznaczności w nim i wpisy do słownika zamian lokalnych trzeba zrobić na podstawie znaczeń wyrazów w kontekście treści pieśni.

4. Na koniec należy dopuścić usuwanie niektórych form podstawowych słów, które nie powinny być indeksowane, gdyż występują nazbyt często, np. spółników czy partykuł. Słowa te zawiera plik `glsout.txt`.
5. Dysponując słownikami zamian można przy pomocy skryptu automatycznie zbudować listę słów dla każdej pieśni i wstawić ją w postaci komend indeksujących do pliku źródłowego.

4.2 Budowa słownika zamian

Pierwszym krokiem jest wyłuskiwanie słów z tekstów pieśni. Można to zrobić skryptem `words.awk` opisanym przy aktualizacji słowników poniżej a następnie przepisywać te słowa krok po kroku do pliku zamian `glsrepl.txt` wraz z dopisaniem w drugiej kolumnie formy podstawowej. Zawartość słownika zamian brana jest pod uwagę przy generowaniu listy słów `words.txt` poprzez zastąpienia oznaczone na liście znakiem „+”.

Można wspomóc się słownikiem form podstawowych ze strony <http://sjp.pl/sownik/odmiany/>, po minimalnej zmianie jego składni, gdyż w pliku `odm.txt` poszczególne formy fleksyjne oddzielane są przecinkami i spacjami. Należy te separatory zastąpić znakami „+”, aby automatyczne wyodrębnianie słów było jednoznaczne. Można to zrobić komendą:

```
awk "{gsub(\" \", \"\\\", \"\\+\\\")};gsub(\"\\[+\\s+\\\", \"\\+\\\")};print}" odm.txt > odpm.txt
```

Słownik tak zmodyfikowany należy zapisać w pliku tekstowym `odpm.txt` kodowanym UTF-8. Teraz możemy się wspomagać skryptem `bases.awk`:

```
{ word=$0;
  if(substr(word,1,1)==""){print;next};
  orgfs=FS; FS="+"; base="";
  fl=substr(word,1,1);
  if(fl~/[A-Z]/){
    fl=tolower(fl); wordlc=fl"substr(word,2)
  }else{
```

```

wordlc=word;
gsub("\xC4\x84","\xC4\x85",wordlc); gsub("\xC4\x86","\xC4\x87",wordlc);
gsub("\xC4\x98","\xC4\x99",wordlc); gsub("\xC5\x81","\xC5\x82",wordlc);
gsub("\xC5\x83","\xC5\x84",wordlc); gsub("\xC3\x93","\xC3\xB3",wordlc);
gsub("\xC5\x9A","\xC5\x9B",wordlc); gsub("\xC5\xB9","\xC5\xBA",wordlc);
gsub("\xC5\xBB","\xC5\xBC",wordlc);
}
file="odmp.txt"; seq=0;
while(getline < file >0){
  if($0~wordlc){
    for(v=1;v<=NF;v++){
      if(length(word)<8){sep="\t\t"}else{sep="\t"}
      if($v==wordlc){if(base!=1){base=$1;if(seq++){print(word"sep"base"seq")
        else{print(word"sep"base)}};break;
      }
    }
  }
}
close(file);
if(length(base)==0){
  base="-"; print(word"\t"base)
}
FS=orgfs;
}

```

wywołanym np. komendą:

```
cat words.txt|awk -f bases.awk
```

który dla podanych w pliku `words.txt` słów nie oznaczonych jeszcze znakiem „+” (a więc jeszcze nie występujących w słowniku zamian) wyszuka w słowniku odmian `odmp.txt` najpierw formę fleksyjną a potem odpowiadającą jej formę podstawową i wypisze obie tak, aby je ewentualnie przepisać do pliku słownika zamian `glsrepl.txt`. W trzeciej kolumnie dodatkowo umieści licznik wersji, gdyż formy fleksyjne mogą odpowiadać wielu różnym formom podstawowym zależnie od semantyki i kontekstu danego słowa.

Po rozbudowaniu słownika zamian można ponownie wygenerować listę słów a następnie ponownie słowa jeszcze nie oznaczone znakiem „+” uzupełniać o formy podstawowe i te zaakceptowane przepisywać do słownika zamian.

Automatyczne uzupełnianie wymaga dla każdej formy fleksyjnej przeglądania całego słownika odmian języka polskiego `odmp.txt` co powoduje, że proces automatycznego uzupełniania słów o formy podstawowe trwa grube dziesiątki minut. Pomimo wspomagania w wyszukiwaniu form podstawowych i tak decyzję o właściwej formie podstawowej albo o niejednoznaczności rozwiązywanej później przez słownik zamian lokalnych musi podjąć człowiek odpowiednio do znaczenia słowa i jego kontekstu.

4.3 Algorytm synchronizacji

Jest także zapisany w AWKu. Sprowadza się do filtrowania kodu `TEX`owego, buforując fragmenty a po osiągnięciu początku nagłówek kolejnej pieśni opróżniając bufor tekstu a podczas tej czynności po osiągnięciu końca nagłówek poprzedniej pieśni przetwarzając jej zbuforowaną odrębnie treść, wyodrębniając listę słów, dokonując zamian na podstawie wczytanych wcześniej słowników

zamian i na koniec wyprowadzając tak uzyskaną listę do pliku roboczego, posortowanie go, usunięcie duplikatów i wczytanie aby następnie wyprowadzić je jako komendy indeksujące zamiast istniejących w tym miejscu podobnych komend. Ostatnio używana wersja tego algorytmu (`songs.awk`) wygląda następująco:

```
BEGIN{while(getline < "glsrepl.txt" > 0){if(length($0))glsrepl[$1]=$2;close("glsrepl.txt");
while(getline < "glsout.txt" > 0){if(length($0))glsout[$1]=1*1;close("glsout.txt");
while(getline < "glsreplpage.txt" > 0){
if(length($0)){repl[$1,$3]=$2;close("glsreplpage.txt");
}
function processsong () {
system("rm song.txt");
for(six=1;six<=sbi;six++){
row=songbuffer[six];
if((substr($0,1,1)!="\045")&&($0!~/songbody/)){
gsub("Š w \\\C i \\\C e t \\\F a","Šw\|C i\|E e\|F a",row);
gsub("\\\\\\K \\\C A \\\E v \\\F e~Mari\\\\\\E a","\\K \|C A\|E v\|F e-Mari\\E a",row);
gsub("\\\\\\ftbackrestore[{}.*[]][{}.*[]]", "",row);
gsub("\\\\\\ftmarksave[{}.*[]]", "",row);
gsub("\\\\\\upflag[{}.*[]]", "",row);
gsub("\\\\\\llap[{}.*[]]", "",row);
gsub("\\\\\\ftmarklast[{}[]]", "",row);
gsub("{[]]", "-",row);
gsub("\\\\\\textbf[{}ci]", "ci",row);
gsub("\\\\\\textbf[{}leñ]", "leñ",row);
gsub("\\\\\\textbf[{}ro]", "ro",row);
gsub("\\\\\\textbf", "",row);
gsub("-*[0-9]*[.]*[0-9][0-9]*em", "",row);
gsub("-*[0-9]*[.]*[0-9][0-9]*ex", "",row);
gsub("\\\\\\\\\\\\\\\\\\", " ",row);
gsub("\\\\\\\\else", " \\\\else",row);
gsub("\\\\\\\\fis", " \\\\Fis",row);
gsub("\\\\\\\\fi", " \\\\fi",row);
row=gensub(/\\{(.*)\\}/, "\\i", "g",row);
row=gensub(/\\((.*)\\)/, "\\i", "g",row);
gsub(" ", " ",row);
gsub(" ", " ",row);
gsub(" ", " ",row);
gsub(" ", " ",row);
gsub(" ", " ",row);
gsub("[A-Za-z]*\\{\\}", " ",row);
gsub("[A-Za-z]*\\\\\\\\", " \\\\",row);
gsub("[A-Za-z]*\\\\\\\\", " \\\\",row);
gsub("[A-Za-z]* ", " ",row);
gsub("[A-Za-z]*", " ",row);
gsub("\\f\\", " ",row);
gsub("\\.\\.", " ",row);
gsub("[,;?;!]", " ",row);
gsub("-", " ",row);
gsub("&", " ",row);
gsub("[0-9=]", " ",row);
gsub("{", " ",row);
gsub("a", " ",row);
gsub("b", " ",row);
gsub(" ", " ",row);
gsub(" ", " ",row);
gsub(" ", " ",row);
gsub(" ", " ",row);
gsub(" ", " ",row);
if(row!~/~/){
split(row,field);
for(i=1;length(field[i])>0;i++){
word=field[i];
gsub("~\342\200\236", "",word);
gsub("~\342\200\235$", "",word);
gsub("\342\200\246$", "",word);
gsub("%$", "",word);
gsub("[\050]", "",word);
gsub("[\051]", "",word);
wordout="";
wordmark="";
if(length(glsrepl[word])>0){
if(glsrepl[word]!="-"){
wordmark="+";wordout=glsrepl[word]
}
}
```

```

        }else{wordout=word};
        if((length(repl[wordout,songid])>0)){
            split(repl[wordout,songid],rs,"&");
            for(rsix=1:length(rs[rsix])>0;rsix++){
                if(rs[rsix]!="-")
                    print(rs[rsix]) > "song.txt";
            }
        }else{if(length(wordout)>0)
            print(wordout) > "song.txt"
        }
    }
}
}
}
close("song.txt");
system("cat song.txt | sort | uniq > songu.txt");
sortedix=0;
while(getline uniqueword < "songu.txt" >0){
    if((length(uniqueword)>0)&&(glsout[uniqueword]!=1)){print("\\index{uniqueword}");
        counters[uniqueword]++;}
}
close("songu.txt");
sbi=0;
print "===done===>songid > "/dev/stderr"
}
function outbuffer () {
    if(bi>1){
        for(ix=1;ix<=bi;ix++){
            print buffer[ix];
            if(buffer[ix]!="\\end{songhead}"){
                processong();
            }
        }
    }
}
{if($0-"\\end{songbody}")songbuffering=0;
if($0-"/\\songid/"){line=$0;gsub("{"," ",line);split(line,a,"");songid=a[2]}
if(buffering==1){if((($0!-"/\\index/)&&($0!-"/copied/))buffer[++bi]=$0;
}
if(songbuffering==1)songbuffer[++sbi]=$0;
if((buffering==0)&&(songbuffering==0)){print $0;
if($0-"\\begin{songhead"){
    outbuffer();
    buffering=1;bi=0
};
if($0-"\\begin{songbody"){songbuffering=1;
}
END{
    outbuffer();
    for(uw in counters){print counters[uw]"\\t"uw > "counters.txt"}
    close("counters.txt");
}

```

Operację tę należy przeprowadzić na każdym pliku źródłowym oddzielnie.

```

cat spliturg.tex | awk -f songs.awk > spliturg.idx.tex
cat sppreka.tex | awk -f songs.awk > sppreka.idx.tex
cat spkatech.tex | awk -f songs.awk > spkatech.idx.tex
cat spwybran.tex | awk -f songs.awk > spwybran.idx.tex
cat spextra.tex | awk -f songs.awk > spextra.idx.tex
cat koledy.tex | awk -f songs.awk > koledy.idx.tex

```

```

copy spliturg.idx.tex spliturg.tex
copy sppreka.idx.tex sppreka.tex
copy spkatech.idx.tex spkatech.tex
copy spwybran.idx.tex spwybran.tex
copy spextra.idx.tex spextra.tex
copy koledy.idx.tex koledy.tex

```

```
cat glsout.txt|awk -f wordlist.awk > glsout.tex
```

Ostatnia operacja przetworzy tak słownik wykluczeń glsout.txt, aby

nadawał się do złożenia pod indeksem w klauzuli wyliczającej słowa wykluczone z indeksu wraz z licznikami ich wystąpień.

4.4 Aktualizacja słowników zamian

W słowniku zamian globalnych `glsrepl.txt` aktualizację w związku z nową pieśnią jest zrobić niełatwo. Możliwe są dwie sytuacje:

1. pojawia się zupełnie nowe słowo z nieistniejącą w słownikach zamian formą podstawową;
2. pojawia się nowa forma fleksyjna słowa, które w formie podstawowej jest już w słowniku zamian globalnych albo lokalnych.

Aby zadanie ułatwić, można posłużyć się skryptyem `words.awk`:

```
BEGIN{while(getline < "glsrepl.txt" > 0){
  if (length(glsrepl[$1])>0){print("?"$1"\t"glsrepl[$1]);glsrepl[$1]=$2;
}
}/begin{songbody/,/end{songbody/{
if((substr($0,1,1)!="045")&&($0!~/songbody/)){
  gsub("Š w \\\C i \\\E ě t \\\F a","Šw\|C i\|E ět\|F a");
  gsub("\\\K \\\C A \\\E v \\\F e-Mari\\\E a","\\K \|C A\|E v\|F e-Mari\|E a,");
  gsub("\\\ftbackrestore[{}.*{}][{}.*{}]", "");
  gsub("\\\ftmarksave[{}.*{}]", "");
  gsub("\\\upflag[{}.*{}]", "");
  gsub("\\\llap[{}.*{}]", "");
  gsub("\\\ftmarklast[{}.*{}]", "");
  gsub("[{} ]", "-");
  gsub("\\\textbf[{}ci{}]", "ci");
  gsub("\\\textbf[{}leñ{}]", "leñ");
  gsub("\\\textbf[{}ro{}]", "ro");
  gsub("\\\textbf", "");
  gsub("-*[0-9]*[.]*[0-9][0-9]*em", "");
  gsub("-*[0-9]*[.]*[0-9][0-9]*ex", "");
  gsub("\\\\\\\\\\\\\\\\\\", " ");
  gsub("\\\\\\\\\\\\\\\\\\", " \\\\else");
  gsub("\\\\\\\\\\\\\\\\\\", " \\\\Fis");
  gsub("\\\\\\\\\\\\\\\\\\", " \\\\fi");
  $0=gensub(/\\((.*)\\)/, "\\1", "g");
  $0=gensub(/\\((.*)\\)/, "\\1", "g");
  gsub(" ", " "); gsub(" ", " "); gsub(" ", " "); gsub(" ", " "); gsub(" ", " ");
  gsub("\\\\\\[A-Za-z]*\\\\\\\\", "\\");
  gsub("\\\\\\[A-Za-z]*\\\\\\\\", "\\");
  gsub("\\\\\\[A-Za-z]*\\\\\\\\", "\\");
  gsub("\\\\\\[A-Za-z]* ", "");
  gsub("\\\\\\[A-Za-z]*", "");
  gsub("\\\\\\\\\\\\\\\\\\", " ");
  gsub("\\\\\\\\\\\\\\\\\\", " ");
  gsub("\\[.]*\\\\", "");
  gsub(", ; ? . ! ", "");
  gsub("-", " ");
  gsub("&", " ");
  gsub("[0-9=]", " ");
  gsub("{", " ");
  gsub("a", " ");
  gsub("b", " ");
  gsub(" ", " ");
  gsub(" ", " ");
  gsub(" ", " ");
  gsub(" ", " ");
  gsub(" ", " ");
  gsub(" ", " ");
  gsub(" ", " ");
  noofwords=NF;
  origfs=FS;
  if($0!~/%/){for(i=1;i<=noofwords;i++){
    word=$1;
    gsub("~\342\200\236", "", word);
    gsub("~\342\200\235$", "", word);
    gsub("~\342\200\246$", "", word);
    gsub("%$", "", word);
    gsub("[\050]", "", word);
    gsub("[\051]", "", word);
    if (length(glsrepl[word])>0){if (length(glsrepl[word])>1)print("+glsrepl[word])}}
```

```

    else {print(word)}
  }
  FS=orgfs;
}
}

```

Skrypt możemy wywołać komendą:

```

cat spliturg.tex sppreka.tex spkatech.tex spwybran.tex spextra.tex koledy.tex \
|awk -f words.awk|sort|uniq >words.txt

```

dzięki czemu otrzymamy plik `words.txt` ze wszystkimi słowami występującymi w tekstach pieśni, przy czym słowa w formach podstawowych zamienionych przez słownik zamian globalnych będą poprzedzone znakiem „+” a zamiany wpisane podwójnie (a więc błędnie, do korekty) będą poprzedzone także znakiem zapytania. Niepoprzedzone żadnym znakiem będą słowa nowe, nieodnalezione w słowniku zamian globalnych.

Oprócz słownika zamian globalnych jest też słownik zamian lokalnych, służący do rozwiązywania niejednoznaczności, `glsreplpage.txt`. Jeśli stosujemy w nim konwencję oznaczania postaci niejednoznacznych znakiem „+”, możemy do wyłowienia takich pozycji indeksowych z identyfikatorami pieśni użyć skryptu AWKowego `glsid.awk`:

```

{ if($0~/\\songid/){
  gsub("{", "");
  split($0,a,"");
  songid=a[2];
}
if($0~/\\index/){
  gsub("{", "");
  split($0,b,"");
  key=b[2];
  if(key~/+/)print(key"\t"key"\t"songid);
}
}

```

Skrypt należy wywołać dla całego tekstu śpiewnika a wynik dopisać do słownika zamian lokalnych, np. komendą:

```

cat spliturg.tex sppreka.tex spkatech.tex spwybran.tex spextra.tex koledy.tex \
|awk -f glsid.awk » glsreplpage.txt

```

po czym ręcznie wyedytować dopisaną część, pozostawiając w drugiej kolumnie dla każdej pieśni wyłącznie pożądaną formę podstawową albo więcej tych form oddzielonych zamiast znaku „+” znakiem „&”.

4.5 Skrypty do generowania indeksu słów

Skrypty działają w środowisku Windows ale przy użyciu narzędzi uniksowych, jak chociażby `awk`, `cat`, `sort` czy `uniq`.

Aby wygenerować komendy indeksujące, należy zadbać o odpowiednią zawartość słowników `glsrepl.txt`, `glsreplpage.txt`, `glsout.txt` oraz wykonać skrypty opisane w rozdziałach 4.5.1 „`words.bat`” i 4.5.2 „`songs.bat`”,

wspomagając się ewentualnie skryptem „`glsid.bat`” z rozdziału 4.5.3. Jeżeli nowych słów pojawiłoby się dużo, można posłużyć się słownikiem odmian i skryptem awkowym opisanym na początku rozdziału 4.2 „Budowa słownika zamian”. Jednak słownik odmian jest stale aktualizowany i bardzo duży (ponad 60 MB), więc nie dołączam go do źródeł. Należy go pobrać samemu, dostosować zgodnie z opisem i dopiero potem posługiwać się skryptem `bases.awk`.

Dla skryptów przetwarzających pliki źródłowe z tekstami pieśni istnieją warianty dla wersji klasycznej i ich odpowiedniki dla wersji alfabetycznej; tem mają na końcu nazwy jedno „s” więcej, np. `songs.bat` — `songss.bat`.

4.5.1 words.bat

Skrypt `words.bat` służy do wybrania słów z kodu L^AT_EXowego śpiewnika i brzmi następująco:

```
cat spliturg.tex sppreka.tex spkatech.tex spwybran.tex spextra.tex koledy.tex ^
|awk -f words.awk|sort|uniq >words.txt
cat spliturg.tex sppreka.tex spkatech.tex spwybran.tex spextra.tex koledy.tex ^
|awk -f words.awk|sort|uniq -c|sort -n -r|awk "{print $2\"\\t-\\\"}" >wordsfreq.txt
cat spliturg.tex sppreka.tex spkatech.tex spwybran.tex spextra.tex koledy.tex ^
|awk -f words.awk|sort|uniq -c|sort -n -r|awk "{print $2\"\\t-\\t\"\\$1}" >wordsfreqs.txt
cat spliturg.tex sppreka.tex spkatech.tex spwybran.tex spextra.tex koledy.tex ^
|awk -f wordsd.awk >wordsd.txt
```

Jak łatwo zauważyć, nad plikami z pieśniami ze wszystkich rozdziałów połączonymi komendą `cat` wykonywany jest skrypt awkowy `words.awk` albo jego wersja raportująca pracę do debugowania `wordsd.awk`.

Skrypt `words.awk` wyluskuje słowa z tekstów pieśni, „oczyszcza” je z komend T_EXowych specyfikujących skład a następnie odszukuje je w słowniku zamian globalnych `glsrepl.txt`. Jeżeli znajdzie, zamienia każde słowo według tego słownika zamian na formę podstawową i wyprowadza poprzedzone znakiem „+”, natomiast jeśli nie znajdzie, wyprowadza takie nowe słowo nie poprzedzone żadnym znakiem. Następnie `sort|uniq` posortuje listę oraz usunie duplikaty. Słowa, na które w `glsrepl.txt` nie ma zamian na wersje podstawowe, znajdują się na końcu listy.

Pierwsza komenda w powyższym skrypcie taką listę słów wyprowadzi do pliku `words.txt`. Zawartość pliku należy zweryfikować a dla nowych słów do słownika zamian globalnych należy dopisać odpowiednie zamiany na ich formy podstawowe.

Mogą się zdarzyć takie słowa, dla których formy podstawowe nie są jednoznaczne. Wtedy należy w słowniku zamian wypisać wszystkie formy podstawowe oddzielone znakiem „+”. Jeżeli zdecydujemy, że słowo nie powinno się znaleźć w indeksie, w drugiej kolumnie zamiast formy podstawowej do słownika zamian globalnych powinniśmy wpisać znak „-”. Tak traktujemy albo se-

kwencje niewyczerpane przez reguły skryptu albo słowa z założenia nieindeksowane (np. komentarze czy niektóre słowa obcojęzyczne). Należy pracować nad uzupełnianiem aż do braku na liście `words.txt` słów niepoprzedzonych znakiem „+”.

Pomocne mogą być dalsze komendy w skrypcie. Do pliku `wordsfreq.txt` przy pomocy parametru `-c` komendy `uniq` oraz dodatkowego przebiegu komendy `sort` uzyskujemy listę słów posortowaną od tych najczęściej występujących. W pliku `wordsfreqs.txt` otrzymamy taką samą listę, tylko w trzeciej kolumnie będziemy dodatkowo mieć licznik powtórzeń słowa. Wreszcie w pliku `wordsd.txt` znajdziemy teksty wiersz po wierszu z poszczególnymi fazami „oczyszczania” tekstu z komend. Plik może być przydatny przy doskonaleniu sekwencji substytucji prowadzących do oczyszczenia tekstu z komend.

4.5.2 songs.bat

Skrypt ten usuwa komendy indeksujące z każdej pieśni i na podstawie wyłuskanych słów generuje nowe komendy indeksujące. Podstawą jest skrypt awkowy `songs.awk` którego przebieg robimy nad każdym plikiem z pieśniami.

```
del glsreplused.txt
cat spliturg.tex | awk -f songs.awk > spliturg.idx.tex
cat sppreka.tex | awk -f songs.awk > sppreka.idx.tex
cat spkatech.tex | awk -f songs.awk > spkatech.idx.tex
cat spwybran.tex | awk -f songs.awk > spwybran.idx.tex
cat spextra.tex | awk -f songs.awk > spextra.idx.tex
cat koledy.tex | awk -f songs.awk > koledy.idx.tex
cat glsreplused.txt | sort | uniq > glsreplUse.txt
```

```
copy spliturg.idx.tex spliturg.tex
copy sppreka.idx.tex sppreka.tex
copy spkatech.idx.tex spkatech.tex
copy spwybran.idx.tex spwybran.tex
copy spextra.idx.tex spextra.tex
copy koledy.idx.tex koledy.tex
```

```
cat glsout.txt|awk -f wordlist.awk > glsout.tex
```

Należy zauważyć, że skrypt awkowy `songs.awk` posługuje się trzema słownikami: globalnym (`glsrepl.txt`) i lokalnym (`glsreplpage.txt`) słownikiem zamian oraz słownikiem słów nieindeksowanych `glsout.txt`. Po przebiegu mamy dodatkowo plik `glsreplUse.txt`, zawierający globalny słownik zamian ale pozbawiony tych wierszy oryginalnego słownika zamian globalnych, które przy przebiegach nie zostały nigdy użyte.

Na koniec przy pomocy skryptu awkowego `wordlist.awk` z pliku słów nieindeksowanych `glsout.txt` oraz z wygenerowanego skryptem `words.bat` pliku `wordsfreqs.txt` tworzy plik `glsout.tex` zawierający słowa nieindekso-

wane wraz z licznikami ich wystąpień. Plik ten używany jest w składzie zaraz pod indeksem słów.

4.5.3 glsid.bat

Skrypt ten w podobnym stylu woła kod awkowy `glsid.awk`.

```
cat spliturg.tex sppreka.tex spkatech.tex spwybran.tex spextra.tex koledy.tex ^
|awk -f glsid.awk >> glsreplpage.txt
```

Tym razem zakłada się jednak, że na plikach z pieśniami został już wykonany skrypt `songs.bat` i już zostały przed tekstami pieśni wstawione komendy `\index` z pozycjami do indeksu słów. Właściwie po poprawnej generacji komend indeksujących wszystko powinno być już dobrze, ale w słowniku zamian globalnych mogły być zadeklarowane niejednoznaczności, czyli kilka form podstawowych oddzielonych znakiem „+”. Te formy dla każdej pieśni oddzielnie powinny zostać zamienione na konkretną formę podstawową według zapisów w słowniku zamian lokalnych `glsreplpage.txt`.

Jeżeli jakaś niejednoznaczność dla jakiejś pieśni nie zostanie przy pomocy słownika zamian lokalnych rozwiązana, w komendzie indeksowej wystąpi ciąg słów ze znakiem „+”. Skrypt awkowy `glsid.awk` wyłapie takie wpisy indeksujące a skrypt `glsid.bat` dopisze je na końcu słownika zamian lokalnych. Teraz trzeba do tego pliku zajrzeć i zobaczyć, czy na jego końcu występują wpisy ze znakiem „+” w drugiej kolumnie. Jeżeli są, to należy je przeredagować, usuwając w drugiej kolumnie niejednoznaczność albo świadomie ją pozostawiając przy użyciu jako separatora znaku „&” zamiast znaku „+” — w takiej sytuacji wszystkie wskazane formy podstawowe zostaną dla danej pieśni zaindeksowane. Następnie należy ponownie uruchomić skrypt `songs.bat` (patrz rozdział 4.5.2) i jeszcze raz `glsid.bat`. Aż do czasu, gdy wszystkie niejednoznaczności zostaną rozwiązane.

Niejednoznaczności mogą się także pojawić w już dopracowanym śpiewniku, w którym zmienimy jakiś identyfikator pieśni i dotychczasowe wpisy do słownika zamian lokalnych nie będą „łapać” niejednoznaczności ze względu na niezmieniony w tym słowniku identyfikator pieśni.

4.6 Generowanie i sortowanie indeksu słów

Podczas kompilacji źródeł komendy `\index{}` wypisują informacje indeksujące uzupełnione o numery stron do pliku `.idx`. Plik ten należy po każdym przebiegu składu przetworzyć zewnętrznym programem aby utworzyć gotowy kod składający indeks w pliku `.ind`. Do konstrukcji indeksu w języku polskim dobrze nadaje się program `texindy`. Aby mógł on pracować w systemie Windows, należy do dystrybucji MikTeXa doinstalować oprogramowanie

Perl, np. w wersji Strawberry. W komendzie wywołania `texindy` należy podać kodowanie UTF-8 oraz język polski parametrami `-C utf8 -L polish`.

5 Warsztat oraz przebiegi kompilacji

Opis będzie dotyczyć systemu operacyjnego Windows, chociaż można wszystko zrobić także w systemie Linux.

Szczegółowa instrukcja przygotowania środowiska pod Windows znajduje się w pliku *InstalacjaŚrodowiska* (w `TeXu` i `PDF-ie`), który jest częścią pakietu dystrybucyjnego i odwołuje się do paczki `Environment.zip`, gotowej do pobrania oddzielnie a zawierającej elementy tego środowiska, sprawdzone i działające pod systemami Windows 10 oraz Windows 11.

5.1 Platforma MiKTeX

Jako platformy `TeXowej` używam `MiKTeXa` <http://miktex.org>.

Przygotowane pliki kompilujemy używając komendy `lualatex SP.tex`. Jeżeli z jakichś powodów budujemy „od zera”, czyli bez plików ze spisem treści, indeksami itd., to do czasu wypisania plików indeksów, przesortowania ich (patrz rozdział następny) oraz ponownego przekompilowania w celu wypisania poprawnego indeksu liter w indeksie alfabetycznym należy kompilować ignorując błędy. Można to uzyskać, odpowiadając `q` na komunikaty albo pierwsze dwa uruchomienia zrobić komendą `lualatex -interaction=nonstopmode SP.tex`. Dopiero, gdy plik `SP.ltr` będzie niepusty, można kompilować bez parametru `-interaction=nonstopmode`.

5.2 Sortowanie indeksów

Początkowo używałem do posortowania wierszy plików indeksowych systemu PowerShell, gdyż tylko `sort` w `powershell-u` uwzględnia kodowanie UTF-8 i polską kolejność alfabetyczną. W końcu pokusiłem się o napisanie własnego programu sortującego w `AWKu`, ze względu na obecność wbudowanej funkcji `asort` z możliwością zadania własnej funkcji porównującej klucze. Ta funkcja w szczególności:

- sama obsługuje kodowanie UTF8,
- nie potrzebuje BOM-u i
- ignoruje w kluczach interpunkcję, dzięki czemu tytuły z przecinkami sortują się prawidłowo;
- wyłuskuje oraz porównuje arytmetycznie ciągi liczbowe, co załatwia sprawę sortowania adresów biblijnych.

Jednak `powershell` sprawia kłopoty z kodowaniem standardowego wyjścia z `AWK-a`, które jest ciągiem bajtów. W efekcie na te potrzeby po wielu

próbach i niespodziankach sprawianych przez kolejne wersje PowerShella zdecydowałem się wywoływać komendę AWK-a ze środowiska CMD. Po tych zmianach skrypt `sortixuni.ps1` zyskał następującą treść:

```
$kind=$args[0]
$sep=$args[1]
$ar=$args[2]
echo "sortix: kind=$kind, sep=$sep, ar=$ar"
#
# sortix =          sortixuni ""          "" <arg>
# sortixkikosq =    sortixuni kikosq ""    <arg>
# sortix-name =     sortixuni ""          "" <name>
# sortixnarrow =    sortixuni narrow ""
# sortixnochords =  sortixuni nochords ""
# sortixpocket =    sortixuni pocket ""
# sortixtab34 =     sortixuni tab34 ""
# sortix-personal = sortixuni <id>        @ <name>
#
$mtx = New-Object System.Threading.Mutex($false, "SortixMutex")
$host.ui.RawUI.WindowTitle = "Wait for $source$kind$sep$ar on Mutex $comment"
If ($mtx.WaitOne(3600000)) {
    $cd=pwd
# Analitic index sort after correction vanishing spaces for bible sort
    cmd /C "awk -f corr.awk $cd\SP$kind$sep$ar.ian|awk -f sort.awk > $cd\SP$kind$sep$ar.ias"
# Alphabetic index sort
    cmd /C "awk -f sort.awk $cd\SP$kind$sep$ar.ial > $cd\SP$kind$sep$ar.ils"
    [void]$mtx.ReleaseMutex()
} else {
    echo "Sortix przeterminowany"
    pause
}
$mtx.Dispose()
```

Aby sortowania nie wchodziły sobie w drogę, są umieszczone w sekcji krytycznej `Mutexa`.

Skrypt ten używa innego skryptu `corr.awk`, który ma korygować pewne słabości informacji indeksowych, i obecnie, po zrobieniu własnego algorytmu sortowania indeksów alfabetycznych i analitycznych, nie zmienia nic. Oto skrypt `corr.awk`:

```
{print}
```

Za sortowanie odpowiada skrypt `sort.awk`:

```
BEGIN{
# lookup table
#
# nawiasy klamrowe w TeX-u nie należą do treści ale mają funkcję organizacyjną;
# zazwyczaj kończącą jakiś fragment tekstu; a ponieważ krótsze powinny być
# wcześniej, nawiasy klamrowe są przed innymi znakami;
# co do nawiasów otwierających nie mam pewności.
#
# nawiasy klamrowe przed spację
c["\x7B"]    =0x20*10+1;#{
c["\x7D"]    =0x20*10+2;#}
#20 - spacja
c["\x20"]    =0x20*10+5
# nieignorowane znaki specjalne
```

```

c["\x2A"]      =0x2A*10+5;##
c["\x2B"]      =0x2B*10+5;#+
#30-39 - 0-9
c["\x30"]      =0x30*10+5;#0
c["\x31"]      =0x31*10+5;#1
c["\x32"]      =0x32*10+5;#2
c["\x33"]      =0x33*10+5;#3
c["\x34"]      =0x34*10+5;#4
c["\x35"]      =0x35*10+5;#5
c["\x36"]      =0x36*10+5;#6
c["\x37"]      =0x37*10+5;#7
c["\x38"]      =0x38*10+5;#8
c["\x39"]      =0x39*10+5;#9
#41-5A - A=Z
c["\x41"]      =0x41*10+5;#A
c["\xC4\x84"]  =0x41*10+6;#A C484
c["\xC3\x81"]  =0x41*10+7;#A C381
c["\x42"]      =0x42*10+5;#B
c["\x43"]      =0x43*10+5;#C
c["\xC4\x86"]  =0x43*10+6;#C C486
c["\x44"]      =0x44*10+5;#D
c["\x45"]      =0x45*10+5;#E
c["\xC4\x98"]  =0x45*10+6;#E C498
c["\xC3\x88"]  =0x45*10+7;#E C388
c["\x46"]      =0x46*10+5;#F
c["\x47"]      =0x47*10+5;#G
c["\x48"]      =0x48*10+5;#H
c["\x49"]      =0x49*10+5;#I
c["\x4A"]      =0x4A*10+5;#J
c["\x4B"]      =0x4B*10+5;#K
c["\x4C"]      =0x4C*10+5;#L
c["\xC5\x81"]  =0x4C*10+6;#L C581
c["\x4D"]      =0x4D*10+5;#M
c["\x4E"]      =0x4E*10+5;#N
c["\xC5\x83"]  =0x4E*10+6;#N C583
c["\xC3\x91"]  =0x4E*10+7;#N C391
c["\x4F"]      =0x4F*10+5;#O
c["\xC3\x93"]  =0x4F*10+6;#O C393
c["\x50"]      =0x50*10+5;#P
c["\x51"]      =0x51*10+5;#Q
c["\x52"]      =0x52*10+5;#R
c["\x53"]      =0x53*10+5;#S
c["\xC5\x9A"]  =0x53*10+6;#S C59A
c["\x54"]      =0x54*10+5;#T
c["\x55"]      =0x55*10+5;#U
c["\xC3\x9A"]  =0x55*10+6;#U C39A
c["\xC3\x9C"]  =0x55*10+7;#U c39C
c["\x56"]      =0x56*10+5;#V
c["\x57"]      =0x57*10+5;#W
c["\x58"]      =0x58*10+5;#X
c["\x59"]      =0x59*10+5;#Y
c["\x5A"]      =0x5A*10+5;#Z
c["\xC5\xBB"]  =0x5A*10+6;#Z C5BB
c["\xC5\xB9"]  =0x5A*10+7;#Z C5B9
#61-7A - a-z
c["\x61"]      =0x41*10+5;#a
c["\xC4\x85"]  =0x41*10+6;#a C485
c["\xC3\xA1"]  =0x41*10+7;#a C3A1
c["\x62"]      =0x42*10+5;#b
c["\x63"]      =0x43*10+5;#c
c["\xC4\x87"]  =0x43*10+6;#c C487
c["\x64"]      =0x44*10+5;#d
c["\x65"]      =0x45*10+5;#e

```

```

c["\xC4\x99"]=0x45*10+6;#e C499
c["\xC3\xA9"]=0x45*10+7;#e C3A9
c["\x66"]      =0x46*10+5;#f
c["\x67"]      =0x47*10+5;#g
c["\x68"]      =0x48*10+5;#h
c["\x69"]      =0x49*10+5;#i
c["\xC3\xAD"]=0x49*10+7;#i C3AD
c["\x6A"]      =0x4A*10+5;#j
c["\x6B"]      =0x4B*10+5;#k
c["\x6C"]      =0x4C*10+5;#l
c["\xC5\x82"]=0x4C*10+6;#i C582
c["\x6D"]      =0x4D*10+5;#m
c["\x6E"]      =0x4E*10+5;#n
c["\xC5\x84"]=0x4E*10+6;#n C584
c["\xC3\xB1"]=0x4E*10+7;#n C3B1
c["\x6F"]      =0x4F*10+5;#o
c["\xC3\xB3"]=0x4F*10+6;#o C3B3
c["\x70"]      =0x50*10+5;#p
c["\x71"]      =0x51*10+5;#q
c["\x72"]      =0x52*10+5;#r
c["\x73"]      =0x53*10+5;#s
c["\xC5\x9B"]=0x53*10+6;#s C59B
c["\x74"]      =0x54*10+5;#t
c["\x75"]      =0x55*10+5;#u
c["\xC3\xBA"]=0x55*10+6;#u C3BA
c["\xC3\xBC"]=0x55*10+7;#ü C3BC
c["\x76"]      =0x56*10+5;#v
c["\x77"]      =0x57*10+5;#w
c["\x78"]      =0x58*10+5;#x
c["\x79"]      =0x59*10+5;#y
c["\x7A"]      =0x5A*10+5;#z
c["\xC5\xBC"]=0x5A*10+6;#z C5BC
c["\xC5\xBA"]=0x5A*10+7;#z C5BA

i=1;
}

function my_utf8_compare(i1, v1, i2, v2, c1, c2, l1, l2, f1, f2, e1, e2,
                        b1, b2, n1, ln1, cn1, n2, ln2, cn2)
{
  gsub(/\\[a-zA-Z]+\|, "", v1);
  gsub(/\\[a-zA-Z]+\|, "", v2);
  #następne bajty są 80-FF
  #C0-DF +1 bajt
  #E0-EF +2 bajty
  #F0-F7 +3 bajty
  #F8-FB +4 bajty
  #FC-FD +5 bajtów
  # non UTF8 version of AWK
  i1=1;i2=1;f1=0;f2=0;e1=0;e2=0;l1=length(v1);l2=length(v2);
  # i1, i2 - lokalne zmienne indeksujące łańcuchy
  # v1, v2 - łańcuchy do porównania
  # c1, c2 - bieżący znak do porównania z pierwszego i drugiego łańcucha
  # l1, l2 - długość pierwszego i drugiego łańcucha
  # b1, b2 - długość znaku utf8 w bajtach w pierwszym i drugim łańcuchu
  # f1, f2 - flagi logiczne znalezienia znaku w tabeli wyszukiwania
  # e1, e2 - flagi logiczne osiągnięcia końca łańcucha
  # n1, n2 - wartość liczby wyłowanej ze stringu
  # ln1, ln2 - długość liczby w znakach
  # cn1, cn2 - kolejny znak liczby
  do {
    while(!f1&&!e1){
      c1=substr(v1,i1);

```

```

if(c1<="\x7F"){
    b1=1;
    ln1=0; n1=0;
    while((i1+ln1)<=l1){
        cn1=substr(v1,i1+ln1,1);
        if(cn1!="0"&&cn1*1==0)break;
        n1=n1*10+cn1;ln1++;
    }
}
else if(c1<"\xC0")b1=1
else if(c1<"\xE0")b1=2
else if(c1<"\xF0")b1=3
else if(c1<"\xF8")b1=4
else if(c1<"\xFC")b1=5;
c1=substr(v1,i1,b1);
if(c1 in c){
    f1=1
}else{
    i1+=b1;
    if(i1>l1)e1=1
}
}
while(!f2&&!e2){
    c2=substr(v2,i2);
    n2=c2=="0"||c2*1>0;
    if(c2<="\x7F"){
        b2=1;
        ln2=0; n2=0;
        while((i2+ln2)<=l2){
            cn2=substr(v2,i2+ln2,1);
            if(cn2!="0"&&cn2*1==0)break;
            n2=n2*10+cn2;ln2++;
        }
    }
    else if(c2<"\xC0")b2=1
    else if(c2<"\xE0")b2=2
    else if(c2<"\xF0")b2=3
    else if(c2<"\xF8")b2=4
    else if(c2<"\xFC")b2=5;
    c2=substr(v2,i2,b2);
    if(c2 in c){
        f2=1
    }else{
        i2+=b2;
        if(i2>l2)e2=1
    }
}
}
if(ln1&&ln2){
    i1+=ln1;
    if(i1>l1)e1=1
    i2+=ln2;
    if(i2>l2)e2=1
    if(n1<n2)return -1
    else if(n1>n2)return 1;
}
if(e1||e2)return 0;
if(c[c1]<c[c2])return -1
else if(c[c1]>c[c2])return 1;

f1=0;f2=0
i1+=b1;
if(i1>l1)e1=1
i2+=b2;

```

```

    if(i2>12)e2=1
  }
  while(!e1&&!e2)
}

{ source[i++]=$0
}

END{
  n = asort(source,dest,"my_utf8_compare")
  for (i = 1; i <= n; i++){
    print dest[i]
  }
}

```

Dlatego potrzebujemy mieć zainstalowany **awk** obsługujący w funkcji **asort** własne funkcje porównujące klucze sortowania. **AWK** przydaje się też do innych rzeczy, np. do składu Nut liturgicznych i śpiewnikowych. Należy podkreślić, że istnieją wersje UTF-8 **AWKa** ale, jak na razie, nie dla Windows. Powyższy algorytm wymaga wersji NIE UTF-8 **AWKa**. Sposób instalacji **AWKa** i innych narzędzi opisany jest w dokumencie **InstalacjaŚrodowiska**.

Sortowanie należy wykonać w środowisku **powershell** -**ExecutionPolicy Unrestricted** wywołując skrypt **.\sortixuni.ps1** z odpowiednimi parametrami opisanymi w skrypcie. Jeżeli będziemy powtarzać budowy, należy rozważyć wykonywanie wszystkich innych operacji także z zachęty **powershella**. Politykę wykonywania skryptów w **powershell**'u można też ustawić globalnie dla całego systemu.

Następnie należy jeszcze raz przekompilować **lualatex SP.tex** a w pliku **SP.pdf** powinniśmy uzyskać złożony Śpiewnik.

Kompilowanie z innymi nazwami — sortowanie Dokładnie te same źródła można zbudować z podaniem innej nazwy pracy (patrz dalej), ale wszystkie pliki pomocnicze, także te z indeksami będą pisane pod innymi nazwami i będą potrzebować do sortowania innych parametrów skryptu sortującego indeksy. Oczywiście można też skopiować nazwę pliku startowego na inną nazwę i zacząć kompilację od tak uzyskanej kopii.

5.3 Budowa nutek

Jedna z pieśni, „Kocham Cię, Panie”, ma załączone nutki instrumentu solowego. Złożone nutki są już skompilowane i najprawdopodobniej nie będzie potrzeby ich zmieniać, gdyż podczas kompilacji Śpiewnika plik **mu.pdf** jest włączany do składu jako ilustracja.

Gdyby jednak zaszła potrzeba jakichś modyfikacji nut w notacji **MusixTeXa**, tekst źródłowy tego składu jest w pliku **mu.tex** ale wymaga on dwuprzebiegowej kompilacji z użyciem systemu **MusixTeX**. Nie chodzi wyłącznie o zestawy makr ale także o program binarny **musixflx.exe**, który

rozkłada na właściwych miejscach nutki w systemach.

W celu ponownego zbudowania pliku `mu.pdf` z nutkami należy wykonać:

```
lualatex -interaction=nonstopmode mu.tex
musixflx.exe mu.mx1
lualatex -interaction=nonstopmode mu.tex
```

Oczywiście po tej budowie należy ponownie skompilować `SP.pdf`, aby zawrzeć w nim odnowione nutki.

Istnieje możliwość włączania kodu MusiX \TeX a wprost do tekstu pieśni. Jednak MusiX \TeX konsumuje dużo zasobów i nie wszystkie kompilatory \TeX a radzą sobie jednocześnie z pakietami `musixtex`, `chords` oraz `spiewnik`. Walka o to rozwiązywanie dawałaby szansę na ewentualne transpozycje zapisu nutowego razem z transpozycją tonacji akordów. Jest to jednak rozwiązanie mało praktyczne, gdyż ta melodia używa najniższego dźwięku skrzypiec i ewentualne transpozycje zapisu nutek w dół czynią je niewykonywalnymi. Po cóż je więc transponować?

5.4 Budowa całości oraz drukowanie

Warto tutaj wspomnieć, że wszystkie formaty budowane są z tego samego źródła, czyli poczynając od pliku `SP.tex`. W prologu zdefiniowane są makra rozpoznające suffiksy w nazwie pracy. Warto przypomnieć, że chociaż nazwa pracy w \TeX u pochodzi z nazwy pliku startowego, to kompilatorowi `luatex` czy `lualatex` można ją podać w parametrze wywołania `--jobname=` niezależnie od nazwy pliku startowego. W zależności od suffiksu nazwy pracy po literach `SP` wybierana jest inna geometria strony oraz ustawiane są dalsze opcje. I tak dla przyrostka:

kikosq — ustawiana jest geometria strony dla wymiaru papieru 21×20 cm; włączany jest dwulamowy skład indeksów z uwzględnieniem skróconych tytułów i podtytułów; Zdefiniowane jest także makro `\Ifsq`, które wykonuje warunkowo albo argument pierwszy albo drugi w zależności od obecności tego suffiksu i używane jest do ustawień innych stopni pisma w takich miejscach jak np. opisy, spisy czy indeksy, gdyż pojemność strony jest inna (powierzchnia większa, strona szersza ale nieco niższa niż A5); w tej wersji składu dodatkowo osadzany jest śpiewnik hiszpański zmodyfikowany poprzez uzupełnienie go o etykiety stron (gdyż ma podobny format, bo 21×21 cm) oraz uaktywniane są łącza od tytułów hiszpańskich wprost do odpowiednich stron tak osadzonego śpiewnika hiszpańskiego;

tab34 — ustawiana jest geometria strony dla wymiaru papieru 15.75×21 cm czyli w proporcjach boków 3×4 , typowej dla czynnika e-papieru i nie-

których tabletów; Zdefiniowane jest także makro `\Iftabw`, które wykonuje warunkowo albo argument pierwszy albo drugi w zależności od obecności tego suffiksu i używane jest do ustawień innych stopni pisma w takich miejscach jak np. opisy, gdyż pojemność strony jest minimalnie większa niż A5;

nochords — geometria strony pozostaje wprawdzie A5 ale zdefiniowane jest makro `\Ifnochords` oraz przy jego pomocy `\ifchords`, które służą do wyłączenia składu akordów oraz do warunkowego składania niektórych pieśni większym stopniem pisma, innego ich łamania oraz wyłączania niektórych przypisów, ściśle związanych z tonacjami i akordami;

pocket — wersja ta przeznaczona jest do wydruku dla braci, którzy chcą używać śpiewnika na przygotowaniach aby wybrać do liturgii odpowiednie pieśni; jest więc jednocześnie monochromatyczna, bez akordów, bez kolęd oraz z dodatkowym marginesem 5mm na oprawę (inne wersje są adresowane albo do użytkowania ich w wersji elektronicznej albo do przechowywania kartek w segregatorze, w odpowiednich koszulkach lub wpinkach samoprzylepnych i marginesu nie potrzebują); wersja **pocket** jest pozbawiona rozdziału z kolędami oraz zamiast kolorowego paska oznaczającego rozdział ma piktogram przy numerze strony;

narrow — to także wersja przeznaczona głównie do wydruku, bo z dodatkowym marginesem 5mm na oprawę; jednak ta odmiana w przeciwieństwie do **pocket**, zawiera cały materiał śpiewnika i jest składana minimalnie węższą odmianą pisma, aby dać miejsce na margines np. do wkręcenia oprawy spiralnej; oczywiście w tej wersji są akordy.

Oprócz przyrostków, określających odmianę śpiewnika, są jeszcze modyfikatory (zaczynają się od myślnika), które dla danej odmiany zmieniają jakiś aspekt związany z jego wyglądem:

-alpha — ten modyfikator wybiera inny układ śpiewnika, zgodny z układem śpiewnika włoskiego „Risuscitò 2020” wziętym z układu śpiewnika hiszpańskiego „Resucitò 2019” i jego poprzedników przy zupełnie nowym przydziale pieśni do rozdziałów oraz z alfabetyczną kolejnością pieśni według tytułów w każdym rozdziale, z wieloma nowymi pieśniami oraz wieloma poprawionymi błędami;

-mono — tak zmodyfikowana nazwa pliku wynikowego będzie złożona (poza okładką) w skali szarości, podobnie jak **pocket**;

-serifchords — normalnie akordy składane są jakąś dostępną pisanką, czyli pismem o kroju przypominającym pismo odręczne; ten modyfikator wymusi składanie akordów drukowanym pismem szeryfowym;

-minion — aby użyć tego modyfikatora, w systemie budowy muszą być zainstalowane kroje pisma Minion/Myriad i CaffischScript Pro. Czcionki są komercyjne, chociaż Minion i Myriad są dystrybuowane z bezpłatnym programem Adobe Acrobat Reader; wtedy ten modyfikator przełączy kroje pism z wolnych Latin Modern, dystrybuowanych z T_EXem na wspomniane.

Zamiast powyższych sufiksów i modyfikatorów można po literach SP w nazwie umieścić:

@Imie-Nazwisko — gdzie Imie-Nazwisko jest identyfikatorem osoby; jeśli ten przyrostek jest obecny, powinien występować jako jedyny i służyć do włączania opcji transpozycji tonacji dla wersji imiennych; Definicje takich sufiksów, osób i powiązania ich ze sobą są w pliku **tps.tex** (o ile istnieje) a warunkowe transpozycje są wykonywane na podstawie komend **\Transposition** z tego pliku, przed wskazanymi w tych komendach pieśniami; dodatkowe opcje dla wersji personalnych są włączane w parametrze opcjonalnym komendy **\PersonName** dla osoby o wskazanym identyfikatorze **Imie-nazwisko** z tego pliku.

Kompilację całości robi skrypt Powershella o nazwie **compile-alluni.ps1**. Aby go wywołać, należy w katalogu z rozpakowanymi plikami źródłowymi wykonać komendę **powershell** a z tak uzyskanej zachęty wykonać komendę wywołującą skrypt **.\compile-alluni.ps1** z odpowiednimi parametrami.

Skrypt ten po zbudowaniu **SP.pdf** zbuduje plik **broszury.pdf** zawierający strony ze **SP.pdf** po dwie A5 na jednej stronie A4 w układach broszurowych z różnymi układami zeszytów. Plik broszur jest przygotowany do druku na drukarce dwustronnej z przekładaniem kartek wzdłuż dłuższego brzegu. Drukarka musi mieć marginesy nie większy niż 5mm. Z pliku należy drukować tylko niektóre strony, te, które uznamy za potrzebne, gdyż w różnych układach te same strony powtarzają się wielokrotnie.

Dla formatu szerszych, niż A5 nie ma możliwości drukowania broszur; należy drukować poszczególne kartki na papierze A4 pionowo z przekładaniem wzdłuż dłuższego boku a potem je obcinać do wysokości 20cm lub 21cm.

W wypadku modyfikacji niektórych aspektów składu należy brać pod uwagę, że wersja **kikosq** ma stronę chociaż szerszą to minimalnie niższą i dlatego dla tej wersji niektóre fonty mają zadeklarowaną inną wielkość.

5.4.1 Marginesy na oprawę

Jak już wspomniałem przy wersji **pocket** oraz **narrow** powyżej, normalnie nie przewidywałem w składzie marginesu na oprawę. Dla wersji 21×21cm marginesy nie są potrzebne nawet, gdy jest używana na papierze, gdyż tradycyjnie kartki (zwykle kartonowe) przechowywane są luzem w futerałach. Gdyby jednak

zaszła taka potrzeba, to można kartki przedziurkować i wpinać w segregator — kartki te są dość szerokie, w pełni nie zagospodarowane tekstem i przy brzegu kartek jest miejsce na dziurki.

Kartki w proporcjach 2×3 , czyli A5, są na tyle wąskie, że niektóre pieśni mieszczą się na nich z trudem i wtedy powierzchnia kartki jest wykorzystana niemalże do samego brzegu. Na ekranie nie przeszkadza to zupełnie. Ale na papierze, szczególnie wtedy, gdy chcemy kartki zbindować, brak marginesu jest uciążliwy. Właśnie dlatego powstały wersje **pocket** oraz **narrow**, które wykorzystując albo nieco więcej miejsca w tekście z powodu rezygnacji z akordów, albo poprzez zastosowanie minimalnie węższego kroju pisma, dodaje 5 mm marginesu na oprawę (bindowanie).

Można też skorzystać z gotowych broszur, które dla wersji A5 bez marginesy przewidują wariant pomniejszony proporcjonalnie o kilka procent (a więc także powiększając marginesy u góry i u dołu), aby wygospodarować margines na oprawę.

5.5 Edycyjne środowisko zintegrowane

Gdyby nasza nas ochota intensywniej popracować na edycją tego, czy innych tekstów w $\text{T}_{\text{E}}\text{X}$ u — $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ u, można rozbudować warsztat o środowisko edycyjne integrujące się z systemem Windows oraz z $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$ em. Dość dobrze do tych celów nadaje się $\text{T}_{\text{E}}\text{X}$ studio, które można pobrać stąd: <http://texstudio.sourceforge.net>.

Ze względu na ciągłą pracę autorów nad poszczególnymi pakietami, należy liczyć się z kłopotami ze składem i koniecznością ich rozwiązywania. Można rozważyć posługiwanie się wcześniejszymi wersjami pakietów i kompilatorów.