Sissejuhatus

Tegemist on C-keele projekti LihtsamLatex dokumentatsiooniga. Et tegu on meie enda välja mõeldud keele kompilaatoriga, siis peaks dokumentatsioon olema põhjalik.

Programmi töötamiseks peab kasutajal installitud olema mõni TEX keele kompilaator, näiteks MiKTEX või muu TEXi kompilaator, mis kasutaks käsku `pdflatex`. Seda käsku kasutab programm, et peale lähtekoodi tõlgendamist ja LATEXi koodi genereerimist see ka kompileerida pdf-failiks. Samuti on tarvilik `config.txt` faili olemasolu programmi enda kaustas ning mingi template faili olemasolu `templates` kaustas.

1 Template failid

Esmalt tegeleme template failidega. Template folder on mõeldud kõikide erinevate template failide hoidmiseks. Hetkel on seal ainult `defaultTemplate.txt`, mille sisu võiks olla järgmine

```
\documentclass{article}
\usepackage{amsfonts}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage[estonian]{babel}

\usepackage{parskip}

\begin{document}
{{content}}
\end{document}
```

Siia saab oma valikul juurde lisada nii palju pakete, kui neid vaja võiks minna. Samuti võib kausta juurde lisada veel template faile, kusjuures nende nimed võivad olla suvalised. Igas template failis peab aga olema märksõna

```
{{content}}
```

mis enamasti käib koostatava dokumendi alustava ja lõpetava definitsiooni vahele. See koht, kus template failis on kirjas `{{content}}`, asendatakse kasutaja kirjutatud lähtekoodifaili tõlkega. Tähendab, lõpuks kompileeritava TEX-faili päis ja jalus on pärit template failist, aga sisuks on kood, mille genereerib programm tõlkides kasutaja kirjutatud lähtekoodi. Seda, kuidas kasutaja kirjutatud lähtekoodifaili tõlgitakse, kontrollib `config.txt` fail.

2 Config.txt

Nüüd saame liikuda järgmise faili juurde. Peamised käskude ja keskkondade definitsioonid, sealjuures ka muud lipud, lähevad kõik `config.txt` faili. Algselt võiks selle sisuks olla

```
template = KAAREL

TEXTMODE KÄSUD
//(arg1)// -> \emph{arg1}
pealk (arg1) -> \secton{arg1}

MATHMODE KÄSUD
sum(al)(ül) -> \sum_{al}^{ül}
to -> \to
inf -> \infty
lim(arg1) -> \lim_{arg1}

KESKKONNAD
enum [multiline, end:{--}] -> \begin{enumerate} #content \end{enumerate} | (item(arg1) -> \item arg1)
```

Välimuseltki on selge, et selle faili sisu on neljas eri osas. Iga osa kirjeldatakse allpool eraldi.

2.1 Template faili nimi

Reaga `template = KAAREL` määratakse template faili nimi, mida kompileerimisel kasutatakse. Selle faili nimi peab olema kirjutatud ilma `.txt` laienduseta. Samanimeline fail peab leiduma kaustas nimega `templates`. Näite

puhul peab leiduma selles kaustas fail `KAAREL.txt`.

2.2 Käsu definitsioonid

Peamine asi, mis meie programmiga teeb LaTeXi kirjutamist kiiremaks on käskude defineerimine. Nendega on võimalik suhteliselt lihtsal moel kirjutatud tekst asendada keerulise LaTeXi koodiga. Neid saabki siinsamas defineerida. Käsu definitsiooni saame jaotada kaheks pooleks

```
Tekst, mida lähtekoodis otsitakse. -> Tekst, mis sellega asendatakse.
```

Niimoodi saame defineerida lihtsad tekstasenduse käsud, näiteks

```
alfa -> \alpha
kord -> \cdot
RR -> \mathbb{R}
```

Vahel sellest aga ei piisa. Peame suutma ka defineerida integraale, summasid ning muid abikäske, mis võtavad endale teatud arv argumente. Selleks saab defineeritavas käsus anda sisse n argumenti, kujul

```
käsuNimi(arg1)[arg2]... -> ℱ(arg1, arg2,...)
```

kus

$$\mathcal{F}(\texttt{arg1, arg2,...})$$

on vastava käsu LaTeXi definitsioon, mis sisaldab endas vajalikke argumente (kusjuures need peavad olema sama nimega, mis käsu definitsiooni vasakul pool). Samuti tuleb eristada käsu defineerimisel argumentide tüüpe. Nendeks on

- `(argumendiNimi)` – Ümarad sulud tähendavad pikemat argumenti. See tähendab, et lähtekoodis argumendi lõppu tähistab ainult tühik või rea lõpp. Sellele vastandub lühem argumenditüüp.
- `[argumendiNimi]` – Kandilised sulud tähendavad lühemat argumenti. See tähendab, et lähtekoodis tähistavad argumendi lõppu peale tühiku ka tähemärgid `+`, `-`, `*`, `=` ja `,`. Tihtipeale ei ole vahet, kas definitsioonis on argumenditüüp märgitud pikaks või lühikeseks. Vahe tuleb aga sisse siis kui tahetakse näiteks kirjutada polünoome. Definitsiooni `^(arg1) -> ^{arg1}` puhul tõlgitakse lähtekoodi tekst `a^n+b^m` koodiks

  ```
  a^{n+b^{m}}
  ```

  Polünoomi kirjutamise soovi jaoks see on vale. Õige tõlge oleks `a^{n}+b^{m}`, mis saavutatakse, kui definitsioonis oleks kasutatud kandilisi sulgusid järgmiselt `^[arg1] -> ^{arg1}`.

Niisiis toome mõned näited

```
sum(al)(ül) -> \sum_{al}^{ül}
   sin(uuga) -> \sin{uuga}
 ^[mingiAsi] -> ^{mingiAsi}
```

Ainus käsk, mida defineerima ei pea, on jagamine, mida automaatselt tõlgitakse kui

```
a/b -> \frac{a}{b}
```

Keskkonna definitsioonid.