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, st. sinna, kuhu tõlge minema peaks.

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 = defaultTemplate

to -> \to
inf -> \infty
lim(arg1) -> \lim_{arg1}

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

Niisiis võimegi toodud faili sisu jaotada kolmeks, millest igaüht vaatleme eraldi.

Üldised lipud. Nendeks on lipud, mida saab defineerida faili alguses ning mis kuidagimoodi mõjutavad programmi tööd. Hetkel on nendeks

- `template` - Määrab ära template faili nime, mida kompileerimisel kasutatakse. Samanimeline fail peab leiduma `template` folderis.

Käsu definitsioonid. Teadaolevalt saab LaTeX koodis kasutada erinevaid käske. 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 -> \textbackslash alpha
kord -> \textbackslash cdot
RR -> \textbackslash mathbb{R}
```

Vahest 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 vajalike 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 `,`. Tightipeale ei ole vahet, kas definitsioonis on argumenditüüp märgitud pikaks või lühikeseks. Vahe tuleb aga sisse siis kui tahetakse näiteks kirjutada polünoome.

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.