A stukicommands csomag

Bereczky Péter
peti.ber007@gmail.com
http://people.inf.elte.hu/berpeti
2018.04.22.

Tartalomjegyzék

Hivatkozások

[1] Lőrentey Károly, stuki.sty, stuki.pdf
Struktogramszerkesztő csomag, és dokumentációja (2001)
http://aszt.inf.elte.hu/∼asvanyi/szd/ (2018-as feltöltési dátummal)

A csomagban szereplő parancsok alapötlete dr. Ásványi Tibor nevéhez fűződik.

1. A stukicommands.sty kiegészítő csomag

1.1. Összefoglalás

A `stukicommands.sty` csomag a `stuki.sty` ([1]) csomag (amely egy egyszerűen használható struktogramszerkesztő környezetet nyújt) segédcsomagja, mely segítségével könnyedén lehet rajzolni LateX környezetben osztálydiagramokat és típusdefiníciókat, illetve a `stuki.sty` csomag kiegészítéseként tartalmazza a számláló és a hátultesztelő ciklus újfajta jelöléseit. Új parancsokat vezet be erre a célra:

- `\NIL` a nullpointer jelölésére használatos parancs
- `\struct{«Osztálynév»}` az osztálydiagram kezdőparancsa
- `\eoStruct` az osztálydiagram lezáró parancsa
- `\typedef{«Típuskonstrukció»}{«Új típusnév»}` a típusdefiníciók leírásához szükséges parancs
- `\for{«Inicializáló utasítás»}{«Ciklusfeltétel»}{«Ciklusmag végén végrehajtandó utasítás»}` a for (számláló) ciklus jelölésére használt parancs
- `\dowhile{«Ciklusfeltétel»}` a hátultesztelő ciklus jelölésére használt parancs
- `\linecomment{«Komment»}` C++-szerű egyszerű, egysoros komment jelölésére használt parancs
- `\multicomment{«Komment»}` C++-szerű egyszerű, többsoros komment jelölésére használt parancs

A csomag használatához szükséges ismerni a `stuki.sty` csomagot, amelyről részletes leírás a [1]-ben található.

1.2. Használata

A csomag használatához a fájlnak (`stukicommands.sty`) a LateX kódot tartalmazó mappában kell lennie. Szintén ebben a mappában kell lennie a `stuki.sty` csomagnak is, mivel ezt egészíti ki, illetve épül rá a stukicommands. A LateX kód elejére a `\usepackage` parancsok közé fel kell venni a stukicommands és a stuki csomagot is: `\usepackage{stuki}`, `\usepackage{stukicommands}`. Ezután már minden készen áll a használathoz.

1.2.1. A NIL parancs

A `\NIL` parancs bárhol használható, mivel egy karakterre vezet be új parancsot, de a LateX kód ezen részének matematikai módban kell tartalmaznia a `\NIL` utasítást.

Minta: $\emptyset$ Kódja: `$\NIL$`

1.2.2. A típusdefiníciók parancsa

A típusdefinícióra vonatkozó parancs egy keretet hoz létre a típusdefiníció számára. Két paramétere van: a meglévő osztály, típus vagy típuskonstrukció neve, illetve az új típusnév.

Például:

typedef Alma **Korte**	Kódja: `\typedef{Alma}{Korte}`

1.2.3. Az osztálydefiníciók parancsa

Az osztálydiagram kezdő parancsa egy paramétert vár, az osztály nevét, amelyre a diagram vonatkozik. Ezt a parancsot követi az adattagok felsorolása láthatósággal opcionálisan, illetve típussal megjelölve (**Konvenció:** alapértelmezetten publikus a láthatóság). Ezután egy `\hline` utasítás következik, majd az osztály metódusainak, függvényeinek felsorolása láthatósággal és típussal opcionálisan megjelölve (konstruktorok, destruktorok és visszatérés nélküli függvények esetében nem szükséges megjelölni a típust). Esetlegesen `\{` és `\}` jelek közé egyszerű implementáció is írható a metódusok, függvények után (fontos, hogy minden sor végére, ami nem parancs, `\\` jelnek kell kerülnie).

Ezek alapján egy UML szabványhoz hasonló osztálydiagram jelenik meg (táblázatokkal megvalósítva).

Példa:

Car
+ Color *color* - Person *owner*
+ Car() + crash(Car *other*) { *color* = $\emptyset$ } - int repair()

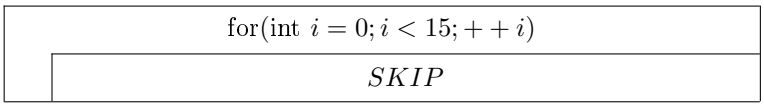
Kódja:

```
\struct{Car}
+ Color $color$ \\
- Person $owner$ \\
\hline
+ Car() \\
+ crash(Car $other$) \{ $color$ = $\NIL$ \} \\
- int repair() \\
\eoStruct
```

1.2.4. A számláló és a hátultesztelő ciklus

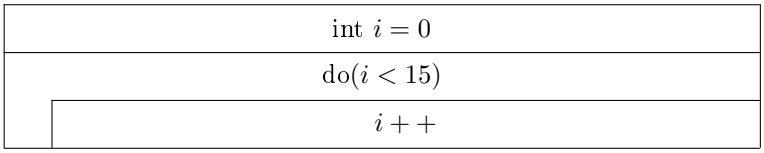
A számláló és a hátultesztelő ciklus jelölésére használatos parancsokra ugyanaz a környezet szükséges (részletek a környezet használatához: [1] 2, 2.4 fejezetei). Struktogram belsejébe kell kerülniük, ezen belül egy `\begin{WHILE}` utasítás ciklusfeltételébe. A számláló ciklusnak három, a hátultesztelőnek egy paramétere van. A számláló ciklus első paramétere a szokásos inicializáló utasítás, a második a ciklusfeltétel, és a harmadik pedig a ciklusmag lefutása után végrehajtandó utasítás (léptető utasítás). A hátultesztelő ciklus egyetlen paramétere a ciklusfeltétel.

Példák:



Kódja:

```
\begin{stuki}
    \begin{WHILE}{1}{\for{$int $i = 0}{i < 15}{++i}}
        \stm{ SKIP }
    \end{WHILE}
\end{stuki}
```



Kódja:

```
\begin{stuki}
    \stm*{int $i = 0$}
    \begin{WHILE}{1}{\dowhile{i < 15}}
        \stm{ i++ }
    \end{WHILE}
\end{stuki}
```

Megjegyzés: Mindkétféle ciklus parancsába írt kifejezés matematikai módban fog értelmeződni a LateX fordító számára. Így pl. a *int* részlet a `\for` parancsban a matematikai mód felfüggesztését jelenti, az „int ” típusnévre nézve (az utána írt szóközzel együtt).

1.2.5. A komment parancsok

A kommentek a kódban bárhol használhatóak, a helyes zárójelezés szabályait figyelembe véve.

Példa:

Car
+ Color *color* //Az autó színe - Person *owner* /*Az autó tulajdonosa*/
+ Car() + crash(Car *other*) { *color* = $\NIL$ }

Kódja:

```
\struct{Car}
+ Color $color$ \linecomment{Az autó színe}\\
- Person $owner$ \multicomment{Az autó \\
tulajdonosa}\\
\hline
+ Car() \\
+ crash(Car $other$) \{ $color$ = $\NIL$ \} \\
\eoStruct
```