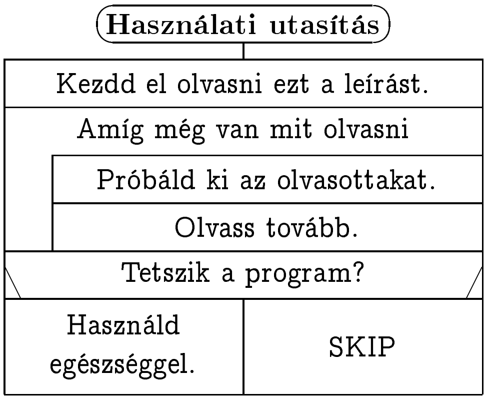
Struktogramok készítése LaTeXben

A *stuki* csomag

Lőrentey Károly
lorentey@inf.elte.hu
http://www.inf.elte.hu/~lorentey/stuki/

2001. január 27.
2.5. változat

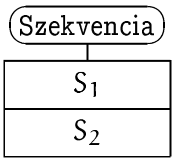


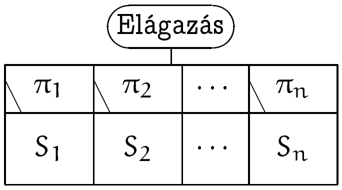
Tartalomjegyzék

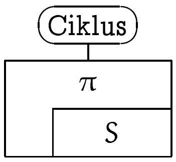
1. Struktogramok

A struktogramok segítségével az algoritmusokat tetszetős, könnyen áttekinthető „dobozok” formájában adhatjuk meg. A struktogramok – nevükhöz méltóan – támogatják, sőt kikényszerítik a strukturált programok írását, és a szintén népszerű pszeudokóddal ellentétben nem kötődnek egyetlen programozási nyelv szintaxisához sem.

Az alapvető programkonstrukciókat a következő módon jelöljük:

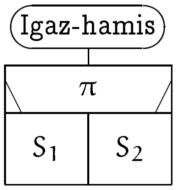






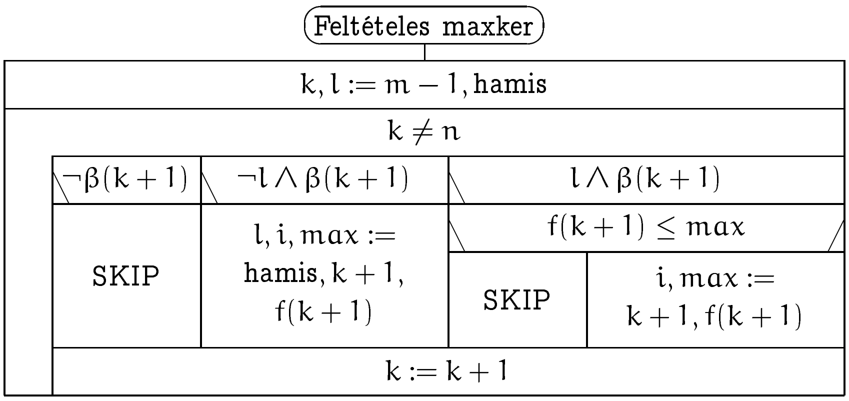
(S tetszőleges struktogramot, π pedig egy logikai kifejezést jelöl.)

Az elágazások egy speciális fajtájára, a kétirányú igaz-hamis elágazásokra olyan gyakran van szükségünk, hogy célszerű számukra bevezetni egy negyedik doboztípust is:



Vegyük észre, hogy az igaz-hamis elágazásdobozok feltételének mindkét szélén van pöcök. A jobboldali pöcök a feltétel hiányzó tagadását kívánja szimbolizálni.

Végezetül adunk egy példát egy valódi, életszagú struktogramra. Íme a feltételes maximumkeresés tételének programja[1]:



Figyeljük meg, hogy az elágazások ágai függőlegesen is szépen középre vannak igazítva.

[1] Ebben a leírásban kizárólag a struktogramok rajzolásával foglalkozunk, így a példaként adott tételek részletes ismertetésétől (állapottér, elő- és utófeltétel, stb.) eltekintünk.

2. A stuki csomag használata

Ebből a fejezetből megtanulhatjuk, hogyan gyárthatunk a LaTeX segítségével a fentiekhez hasonló szép struktogramokat.

2.1. Kezdőlépések

Ha egy dokumentunkban struktogramokat szeretnénk rajzolni, akkor a `stuki` csomagot tartalmazó `stuki.sty` fájlt be kell másolnunk a dokumentumot tartalmazó könyvtárba, és a preambulumba (a dokumentum `\begin{document}` előtti részébe, a `\documentclass` parancs után) be kell írnunk a következő utasítást:

```
\usepackage{stuki}
```

Ezzel a dokumentumunk belsejében elérhetővé tesszük a struktogramgeneráló parancsokat.

2.2. A stuki és a stuki* környezetek

A `stuki` csomag címkézett és címke nélküli struktogramok készítésére is használható. Címke nélküli stukikat a `stuki` környezetben definiálhatunk. A környezetnek egyetlen, opcionális paramétere van, a leendő stuki szélessége. Ha ezt nem adjuk meg, akkor az alapértelmezett szélesség a `\stukiwidth` hosszúságparancs értéke, ami kezdetben 10 cm, de a `\setlength` parancs segítségével bármikor átállíthatjuk:

```
\begin{stuki}[3cm]
  \stm*{Szia, vil\'ag!}
\end{stuki}
```

Szia, világ!

```
\setlength{\stukiwidth}{4cm}
\begin{stuki}
  \stm*{Szia, VIL\'AG!}
\end{stuki}
```

Szia, VILÁG!

A `stuki` környezetnek van egy csillagos változata is, a `stuki*`. Ez csak abban különbözik a stukitól, hogy van egy kötelező paramétere is, a stuki neve, amit a stuki fölé egy lekerekített dobozba ír bele:

```
\begin{stuki*}[3cm]{Els\H o}
  \stm*{Szia, vil\'ag!}
\end{stuki*}
```

Első

Szia, világ!

Néha szükségünk lehet arra, hogy néhány struktogramot egymás mellett jelenítsünk meg (lásd például az 1. oldalon látható stukikat). Erre a feladatra a fenti környezetek nem alkalmasak, hiszen ezek a struktogramokat mindig külön sorokba teszik. A megoldást a `stukibox` és a `stukibox*` környezetek szolgáltatják[2]. Ezek ugyanúgy működnek, mint az eddig megismert társaik, de az eredményüket egy ún. *hbox*-ban adják vissza. A `stukibox` környezetet használva könnyedén rajzolhatunk egymás melletti stukikat:

```
\noindent\hfill
\begin{stukibox*}[3cm]{1. stuki}\stm*{Micimackó}\end{stukibox*}%
\hfill
\begin{stukibox*}[3cm]{2. stuki}\stm*{Malacka}\end{stukibox*}%
\hfill{}
```

1. stuki	2. stuki
Micimackó	Malacka

2.3. Egyszerű utasítások

A fenti példákban már láthattuk, hogy a stukirajzoló környezetek belsejébe az `\stm*` paranccsal tetszőleges szöveget tehetünk. Ez a parancs az `\stm` parancs speciális alakja. A két parancs mindössze abban különbözik, hogy míg az `\stm` matematikai módban jeleníti a paraméterét, addig az `\stm*` sima szöveges módban dolgozik. (A szokásos struktogramokban az utasítások óriási többsége `\stm`, a csillagos változatra csak ritkán van szükség.)

```
\begin{stuki}[3cm]
  \stm{x := \phi(y^2)}
\end{stuki}
```

$x := \phi(y^2)$

Az utasítások szekvenciáit egyszerűen a parancsok egymás után írásával kapjuk:

```
\begin{stuki}[4cm]
  \stm*{Kinyitom az ajt\'ot}
  \stm*{Kiveszem a zsir\'afot}
  \stm*{Beteszem az elef\'antot}
  \stm*{Becsukom az ajt\'ot}
\end{stuki}
```

Kinyitom az ajtót
Kiveszem a zsiráfot
Beteszem az elefántot
Becsukom az ajtót

Figyeljük meg, hogy az elválasztóvonalak automatikusan megjelennek.

Néha egy utasítás nem fér el egy sorban. Ekkor a `\\` paranccsal sortöréseket kell helyeznünk az `\stm` argumentumába[3]. Ha azonban ezt tesszük, akkor az utasítás (sorok számában

[2] Ezek a környezetek a 2.0-ásnál régebbi `stuki` változatokból hiányoznak, de ott a `stuki` környezetek nem is rakják külön sorba a megrajzolt struktogramot.

[3] A `stuki` csomag 1.3-nál korábbi változataiban ügyelni kellett arra, hogy a többsoros `stm*` parancsokban a \\ sortörések előtt ne legyen szóköz. Azóta a szerző egy kicsit TEXnikásabb lett, és ezt a hiányosságot ki tudta javítani.

mért) magasságát szögletes zárójelek között meg kell adnunk.

```
\begin{stuki}[4cm]
  \stm*[2]{Hull a h\'o \'es h\'ozik\\
          zik-zik}
  \stm*[2]{Micimack\'o f\'azik\\
          zik-zik}
\end{stuki}
```

Hull a hó és hózik zik-zik
Micimackó fázik zik-zik

Nem jelent hibát, ha máskor is megadjuk ezt az opcionális magasságértéket, hiszen ez tulajdonképpen az utasításdoboz magasságát adja meg. Ha az utasítás szövege nem tölti ki a megadott magasságot, akkor az `\stm` parancs középre igazítja a szöveget, azaz a hiányzó részt alul-felül egyforma magasságú láthatatlan térközzel tölti ki.

Ha viszont az utasítás magasságánál kisebb értéket adunk meg az utasításdoboz magasságának (vagy nem adunk meg értéket, ha az utasítás nem fér ki egy sorban), akkor a szöveg ki fog lógni a dobozából, és ez nagyon ronda struktogramokat eredményez. Szerencsére ilyen esetekben a LaTeX figyelmeztető üzenetet (`"Overfull \vbox"`) ad. Ha a dokumentum fordítása közben ilyen üzenetet kapunk, akkor valószínűleg valamelyik utasításdoboznak elfelejtettünk magasságértéket adni.

Néha szükségünk lehet nem egész magasságú utasításdoboz létrehozására. Ilyenkor az utasítás magasságaként egyszerűen beírjuk a kívánt valós számot:

```
\begin{stuki}
  \stm*[1]{Dobozolni j\'o!}
  \stm*[1.42]{Dobozolni j\'o!}
  \stm*[1.84]{Dobozolni j\'o!}
\end{stuki}
```

Dobozolni jó!
Dobozolni jó!
Dobozolni jó!

2.4. Ciklusok

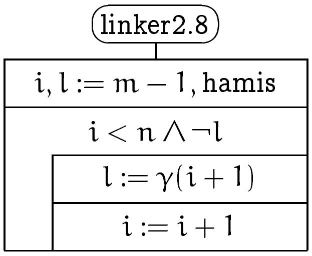
Eljött az ideje, hogy elkezdjünk foglalkozni a bonyolultabb programkonstrukciók ábrázolásával. A szekvencia leírása – mint azt már tudjuk – egyszerűen a komponensek egymásutánírásával történik. A ciklus leírása már valamivel bonyolultabb, ehhez egy új környezettel kell megismerkednünk.

A `WHILE` környezet két kötelező és egy opcionális paraméterrel rendelkezik.

- *Az első kötelező paraméter* az utasítások magasságparaméterének felel meg, megadja a ciklus magjának magasságát. A legegyszerűbb esetben ez a ciklusmag utasításainak száma.
- *A második kötelező argumentumban* kell megadnunk a ciklusfeltételt. A feltételt is utasításként, azaz egy `\stm` vagy `\stm*` parancsként kell beírni.
- *Opcionális paraméterként* egy LaTeX hosszt írhatunk be, ez szabályozza a ciklusmag indentálásának mértékét (alapértelmezésben ez a `\loopindent` makró értéke, ami kezdetben megegyezik az egységsor magasságával.

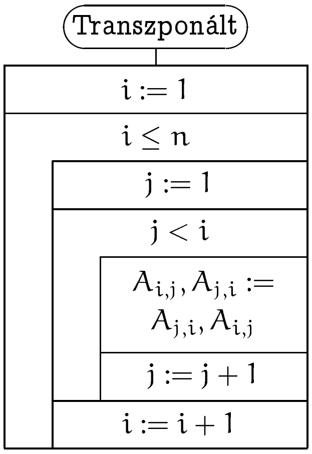
A környezet testébe a ciklusmag utasításait írhatjuk be. Példaként tekintsük a *lineáris keresés intervallumon* programot:

```
\begin{stuki*}{linker2.8}
  \stm{i,l:=m-1,\hbox{hamis}}
  \begin{WHILE}{2}{\stm{i<n\land\lnot l}}
    \stm{l:=\gamma(i+1)}
    \stm{i:=i+1}
  \end{WHILE}
\end{stuki*}
```



Természetesen ciklusokat (elágazásokat stb.) akármilyen mélységben egymásbaágyazhatunk, csak figyelnünk kell a külső ciklus magasságértékére: egy ciklus teljes magassága pontosan a ciklusfeltétel és a ciklusmag magasságainak összege. (Ez a következő két fejezetben bemutatott elágazásokra is igaz: egy elágazás magassága a feltétel és a közös ágmagasság összege.)

```
\begin{stuki*}{Transzpon\'alt}
  \stm{i:=1}
  \begin{WHILE}{6}{\stm{i\le n}}
    \stm{j:=1}
    \begin{WHILE}{3}{\stm{j<i}}
      \stm[2]{
        A_{i,j},A_{j,i}:=\\
        A_{j,i},A_{i,j}}
      \stm{j:=j+1}
    \end{WHILE}
    \stm{i:=i+1}
  \end{WHILE}
\end{stuki*}
```



2.5. Igaz-hamis elágazások

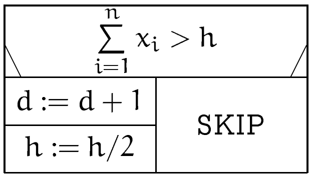
Az igaz-hamis elágazásokat az `IF` nevű környezet segítségével hozhatjuk létre. A környezetnek két kötelező és egy opcionális paramétere van:

- *Az első kötelező paraméter* az elágazás feltétele alatti programrész magasságát határozza meg, a szokásos módon.
- *A második kötelező paraméter* az elágazás feltétele. A feltétel utasításnak számít, tehát `\stm` vagy `\stm*` parancsként kell megadni.
- *Az opcionális paraméter* az első ág szélességének és a teljes elágazás szélességének az arányát adja meg, százalékban. Ha nem adjuk meg, az alapértelmezés 50%, azaz a két ág fele-fele arányban osztozik a rendelkezésre álló területen.

A környezet belsejébe a ágak utasításait kell írni, először az „igaz” ágat, aztán tőle egy \ELSE paranccsal elválasztva a hamisat.

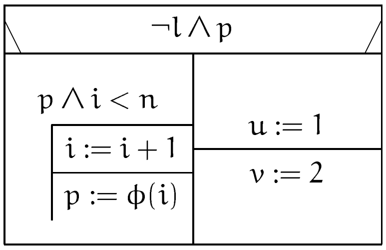
Például tekintsük a következő programot:

```
\begin{stuki}
  \begin{IF}{2}
        {\stm[1.5]{\sum\limits
         _{i=1}^n x_i>h}}
    \stm{d:=d+1}
    \stm{h:=h / 2}
  \ELSE
    \stm*{SKIP}
  \end{IF}
\end{stuki}
```



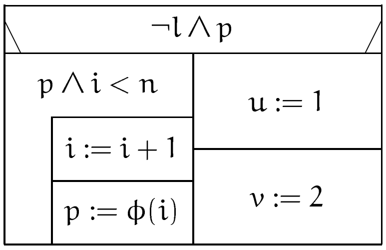
Látható, hogy nem kell megijednünk, ha egy elágazás feltétele magasabb a szokásosnál, a LaTeX nem jön tőle zavarba. Azt is megfigyelhetjük, hogy a SKIP ág anélkül is szép lett, hogy az \stm* parancsnak megadtuk volna az ág kívánt magasságát. Ez annak következménye, hogy az elágazás ágai maguk is teljes egészében középre vannak igazítva, így a fenti esetben az egy magasságú utasítás felett és alatt fél-fél sor magasságú láthatatlan térközök vannak. (Az utasításokat elválasztó vonalak most hiányoznak, hiszen az ág egyetlen utasításból áll.) Az ág magasságát kizárólag akkor lehet büntetlenül figyelmen kívül hagyni, ha az ág csak egyetlen, egyszerű utasítást tartalmaz. A következő példa megmutatja, hogy milyen csúnya eredményt kaphatunk, ha az elágazáson belül óvatlanul elhagyjuk a magasságértékeket:

```
% Hibas!
\begin{stuki}[5cm]
  \begin{IF}{4}{\stm{\lnot l\land p}}
    \begin{WHILE}{2}{\stm{p \land i<n}}
      \stm{i:=i+1}
      \stm{p:=\phi(i)}
    \end{WHILE}
  \ELSE
    \stm{u:=1}
    \stm{v:=2}
  \end{IF}
\end{stuki}
```



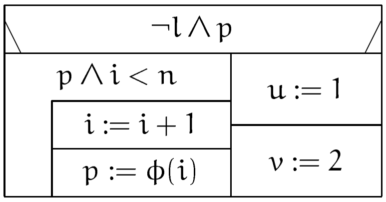
A javítás kézenfekvő: (Figyeljük meg a ciklus magasságértékeit!)

```
\begin{stuki}[5cm]
  \begin{IF}{4}{\stm{\lnot l\land p}}
    \begin{WHILE}{2.66}
            {\stm[1.34]{p \land i<n}}
      \stm[1.33]{i:=i+1}
      \stm[1.33]{p:=\phi(i)}
    \end{WHILE}
  \ELSE
    \stm[2]{u:=1}
    \stm[2]{v:=2}
  \end{IF}
\end{stuki}
```



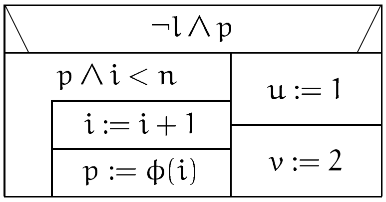
Természetesen a legjobb eredményt úgy kapjuk, ha az elágazás magasságát az ágaknak szükséges magasságok maximumára állítjuk, és a ciklust tartalmazó, helyigényes ágat egy kicsit kiszélesítjük:

```
\begin{stuki}[5cm]
  \begin{IF}[60]{3}
            {\stm{\lnot l\land p}}
    \begin{WHILE}{2}{\stm{p \land i<n}}
      \stm{i:=i+1}
      \stm{p:=\phi(i)}
    \end{WHILE}
  \ELSE
    \stm[1.5]{u:=1}
    \stm[1.5]{v:=2}
  \end{IF}
\end{stuki}
```



Az elágazásfeltételek szélén megjelenő ferde vonalak – „pöckök” – magasságegységekben kifejezett méretét a `\slashheight` makró adja meg. Ez alapesetben 0.666, azaz a pöckök egy egységnyi magasságú feltételdoboznak körülbelül a kétharmadáig ér fel. A makró felüldefiniálásával ez az érték a felhasználó ízléséhez igazítható. A következő stukiban például a pöckök az egységdoboz tetejéig felérnek:

```
\renewcommand{\slashheight}{1}% <===
\begin{stuki}[5cm]
  \begin{IF}[60]{3}
            {\stm{\lnot l\land p}}
    \begin{WHILE}{2}{\stm{p \land i<n}}
      \stm{i:=i+1}
      \stm{p:=\phi(i)}
    \end{WHILE}
  \ELSE
    \stm[1.5]{u:=1}
    \stm[1.5]{v:=2}
  \end{IF}
\end{stuki}
```



Kis léptékű stukik szedésekor előfordulhat, hogy a LaTeX a dokumentum fordítása közben „`\oval, \circle, or \line size unavailable`" figyelmeztetéseket ad, és a pöckök egyszerűen eltűnnek a stukik elágazásairól. Ez a LaTeX `picture` környezetének a hiányossága, melyet legegyszerűbben a `eepic` vagy a `pspicture` makrócsomagok betöltésével oldhatunk meg (`\usepackage{eepic}`). Ezt a `stuki.sty` azért nem teszi meg automatikusan, mert az esetek többségében teljesen feleslegesen terhelné vele a felhasználó türelmét és a rendszer erőforrásait.

2.6. Általános, többágú elágazások

A `stuki` csomag legbonyolultabb része a többágú elágazásokat rajzoló `CASE` környezet. A `CASE` környezet az `IF`-hez hasonlóan két kötelező és egy elhagyható paraméterrel rendelkezik, azonban a paraméterek jelentése más:

- *Az első kötelező argumentumban* az eddigiekhez hasonlóan az elágazás ágainak magasságát kell megadni. Ebbe a feltételek magassága nem számít bele.
- *A második kötelező argumentumban* – egyszerűen fogalmazva – az *ágak számát* kell megadnunk, egész számként. Valójában a LaTeX ezt a számot az *egységnyi ágszélesség* meghatározására használja úgy, hogy leosztja vele az elágazás teljes szélességét.
- *Az opcionális paraméter* a feltételsor magasságát tartalmazhatja. Akkor kell megadni, ha van egynél magasabb feltételdobozú elágazáság. (A feltételek sorát alkotó utasításoknak nem feltétlenül kell egyforma magasnak lenniük, a rövidebbeket a LaTeX a szokott módon középre igazítja. Ehhez azonban előre ismernie kell a legnagyobb utasítás magasságát.)

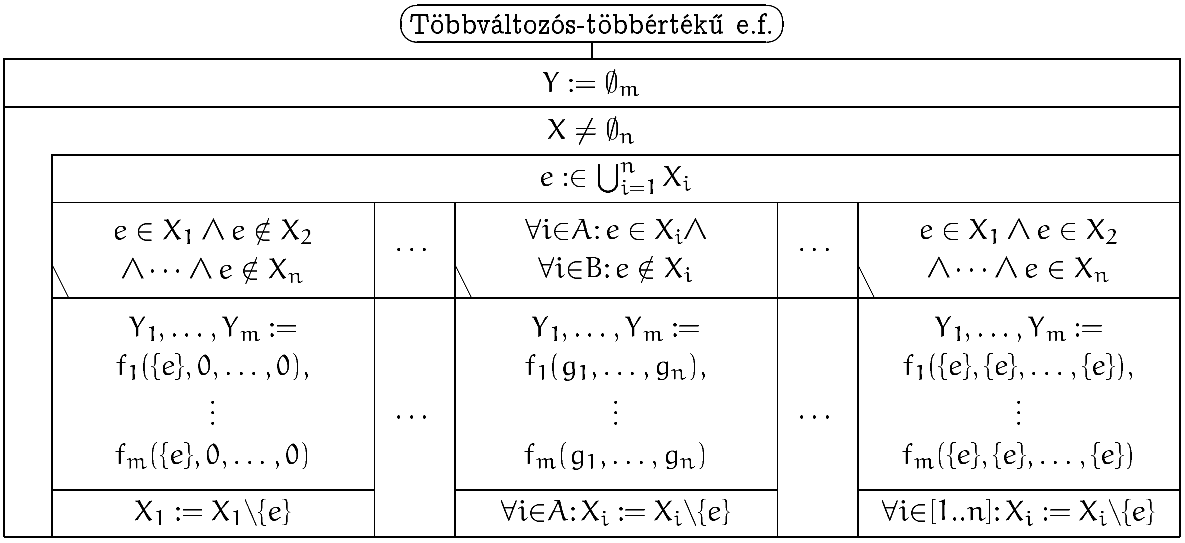
Az elágazás ágait a `CASE` környezet belsejében, `\WHEN` parancsokkal kezdődő utasítássorozatokként adhatjuk meg. A `\WHEN`-nek egy kötelező és egy opcionális paramétere van:

- *A kötelező argumentumban* az adott ág feltételét kell megadni, a szokott módon egy `\stm` parancsba zárva.
- *Az opcionális paraméter* az ág relatív szélességét határozza meg. Az ág tényleges szélességét a relatív szélesség és az egységnyi ágszélesség összeszorzásával kaphatjuk meg. (Értelemszerűen az ágak relatív szélességeinek összege meg kell, hogy egyezzen a `CASE` környezet második argumentumában megadott értékkel.) Ha nem adjuk meg, a paraméter alapértelmezett értéke 1.

A relatív szélességekkel játszi könnyedséggel adhatjuk meg az elágazás ágainak szélességét. (Ha valaki szeret százalékokban számolni, akkor a környezet második paraméterének 100-at adva az egyes ágak szélességét a teljes szélesség százalékában fejezheti ki.)

A `\WHEN` parancsnak van egy csillagos változata is. A `\WHEN*` ugyanúgy működik, mint a sima `\WHEN`, de az általa definiált ág feltételdobozába nem rak pöcköt. Ez programsémák esetében hasznos, amikor nem ismerjük az ágak pontos számát.

Példaként lássuk az általános elemenkénti feldolgozás programját: (Most is csak a struktogramot rajzoljuk fel, a specifikációtól és a felhasznált szimbólumok (A, B, g_i) definíciójától eltekintünk.)



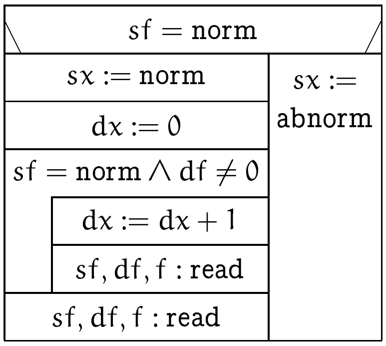
A struktogramot generáló `stuki` környezet a következő oldalon található. Első pillantásra a forráskód riasztóan bonyolultnak tűnhet, de valójában egyszerű szerkezetű. A kód legnagyobb részét az utasítások leírása teszi ki.

```
\begin{stuki*}[\textwidth]{T\"obbv\'altoz\'os-t\"obb\-\'ert\'ek\H u e.f.}
  \stm{Y:=\emptyset_m}
  \begin{WHILE}{8}{\stm{X\ne\emptyset_n}}
    \stm[1]{e:\in\bigcup_{i=1}^n X_i}
    \begin{CASE}[2]{5}{14}
      \WHEN[4]{\stm[2]{e\in X_1\land e\notin X_2\\
                       \land\cdots\land e\notin X_n}}
        \stm[4]{Y_1,\dotsc,Y_m:=\\
                f_1(\{e\},0,\dotsc,0),\\
                \vdots\\
                f_m(\{e\},0,\dotsc,0)}
        \stm{X_1:=X_1\backslash\{e\}}
      \WHEN*[1]{\stm[2]{\cdots}}
        \stm{\cdots}
      \WHEN[4]{\stm[2]{\forall i\mathord\in
                        A\mathpunct: e\in X_i\land\\
                       \forall i\mathord\in
                        B\mathpunct: e\notin X_i}}
        \stm[4]{Y_1,\dotsc,Y_m:=\\
                f_1(g_1,\dotsc,g_n),\\
                    \vdots\\
                f_m(g_1,\dotsc,g_n)}
        \stm{\forall i\mathord\in A\mathpunct:
                      X_i:=X_i\backslash\{e\}}
      \WHEN*[1]{\stm[2]{\cdots}}
        \stm{\cdots}
      \WHEN[4]{\stm[2]{e\in X_1\land e\in X_2\\
                       \land\cdots\land e\in X_n}}
        \stm[4]{Y_1,\dotsc,Y_m:=\\
                f_1(\{e\},\{e\},\dotsc,\{e\}),\\
                \vdots\\
                f_m(\{e\},\{e\},\dotsc,\{e\})}
        \stm{\forall i\mathord\in [1..n]\mathpunct:
               X_i:=X_i\backslash\{e\}}
    \end{CASE}
  \end{WHILE}
\end{stuki*}
```

2.7. A középre igazítás megszűntetése

Ha egy elágazáságban felülre akarjuk igazítani az utasításokat, akkor az adott ág első `\stm` parancsa elé írjunk be egy `\vfilneg` parancsot (ez csak akkor hatásos, ha az ág nincs teljesen kitöltve):

```
\begin{stuki}[5cm]
  \begin{IF}[70]{6}{\stm{sf=\hbox{norm}}}
     \stm{sx := \hbox{norm}}
     \stm{dx := 0}
     \begin{WHILE}{2}{\stm{sf=\hbox{norm}
            \land df\ne0}}
        \stm{dx := dx + 1}
        \stm{sf,df,f:\hbox{read}}
     \end{WHILE}
     \stm{sf,df,f:\hbox{read}}
  \ELSE
     \vfilneg
     \stm[2]{sx :=\\ \hbox{abnorm}}
  \end{IF}
\end{stuki}
```



(Ha alulra akarunk igazítani, akkor a `\vfilneg`-et értelemszerűen az ág utasításai után kell írni.)

3. A `stuki` csomag konfigurációja

Ebben a fejezetben összefoglaljuk, hogy milyen lehetőségeink vannak a `stuki` csomag testreszabására.

3.1. Egyszerű beállítások

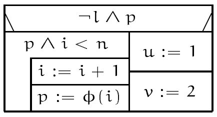
A struktogramok kinézetét néhány egyszerű parancs segítségével könnyedén befolyásolhatjuk:

- A struktogramok alapértelmezett szélességét a `\stukiwidth` hosszparancs tárolja. Kezdeti értéke 10 cm. A 2.2. fejezetben, a 2. oldalon adtunk egy példát a használatára.
- Az utasítások egységnyi magasságát az `\stmheight` hosszparancs adja meg. A struktogramok definiálásakor minden magasságérték ennek a hossznak valamilyen többszöröse. Alapértelmezett értéke 18 pt. (Ha 12 pontnál kisebbre állítjuk, akkor át kell definiálnunk az elágazások pöckeit is (lásd `\stuki@rightif` és `\stuki@leftif` makrók, 3.2 fejezet), különben kilógnának a feltételdobozokból.)
- Az elágazáságak pöckeinek az egységmagassághoz viszonyított méretét a `\slashheight` makró adja meg. A részleteket a 2.5. oldalon megbeszéltük.

- A ciklusmagok alapértelmezett indentálását a `\loopindent` parancs átdefiniálásával változtathatjuk meg. A parancs kezdeti definíciójának értéke megegyezik az éppen aktuális sormagassággal. (Ritkán kell megváltoztatni, hiszen a behúzás mértékét egyedi esetekben a `WHILE` környezet opcionális argumentumában könnyen felül lehet bírálni.)
- A `\thicklines` parancs kiadásával megnövelhetjük a dobozok keretének a vastagságát. Az eredeti állapotot a `\thinlines` parancs állítja vissza. A vastagabb dobozokhoz félkövér betűtípus illik.

Az `\stmheight` parancsot akkor érdemes használni, ha a szokásostól eltérő méretű stukikat szeretnénk rajzolni: (Hasonlítsuk össze ezt a struktogramot a 7. oldalon látható eredetivel!)

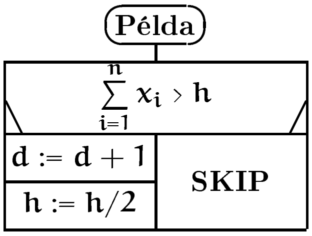
```
\begin{tiny}
 \setlength{\stmheight}{10pt}
 \begin{stuki}[2.75cm]
   \begin{IF}[60]{3}
            {\stm{\lnot l\land p}}
     \begin{WHILE}{2}{\stm{p \land i<n}}
       \stm{i:=i+1}
       \stm{p:=\phi(i)}
     \end{WHILE}
   \ELSE
     \stm[1.5]{u:=1}
     \stm[1.5]{v:=2}
   \end{IF}
 \end{stuki}
\end{tiny}
```



A fenti struktogram elágazásában a pöckök már túl aprók ahhoz, hogy a sztenderd LaTeX megbírkózzon velük, ezért kénytelenek voltunk a dokumentum preambulumában betölteni a `eepic` makrócsomagot.

A `\thicklines` parancs használatára is mutatunk egy példát:

```
\begingroup
 \thicklines
 \mathversion{bold} \bfseries
 \begin{stuki*}{P\'elda}
   \begin{IF}{2}
         {\stm[1.5]{\sum\limits
          _{i=1}^n x_i>h}}
     \stm{d:=d+1}
     \stm{h:=h / 2}
   \ELSE
     \stm*{SKIP}
   \end{IF}
 \end{stuki*}
\endgroup
```



3.2. Érdekes belső parancsok

Az alábbi parancsok a `stuki` csomag a konfiguráció szempontjából érdekes belső parancsai. Különleges beállítási igények esetén szükség lehet a megváltoztatásukra, de a csomag normális használatához nem szükséges az ismeretük.

- A `\stuki@hrule` és `\stuki@vrule` parancsok nulla látszólagos vastagságú, de maximális hosszúságú vízszintes, illetve függőleges vonal húzására szolgálnak. A stukik dobozai ilyen vonalakból épülnek fel. A vonalak tényleges vastagsága alapértelmezésben 0.4 pt, ez a `\thicklines` parancs kiadásával 1 pt-ra változtatható. Az ezektől különböző vonalvastagságokat a két parancs átdefiniálásával lehet beállítani (erre gyakorlatilag soha sincs szükség).
- A `\stuki@leftif` és `\stuki@rightif` parancsok az elágazások pöckeinek megrajzolására szolgálnak. A rajz nem rendelkezhet nullánál nagyobb (látszólagos) méretekkel.
- A `\stuki@title` parancs a `stuki*` környezet ovális címkedobozait rajzolja. Speciális effektusok létrehozásához ezt a makrót is átdefiniálhatjuk. A parancs két paramétert kap, a stuki teljes szélességét és a dobozba teendő szöveget.

Összefoglaló a **stuki.sty** csomagról

Utasítások

`\begin{`**stuki**`}[`*szélesség*`]`	Egyszerű stukik
`\begin{`**stuki***`}[`*szélesség*`]{`*címke*`}`	Címkézett stukik
`\begin{`**stukibox**`}[`*szélesség*`]`	Egyszerű stuki, hboxban
`\begin{`**stukibox***`}[`*szélesség*`]{`*címke*`}`	Címkézett stuki, hboxban
\stm`{`*képlet*`}`	Utasítás matematikai módban
\stm*`{`*szöveg*`}`	Szöveges utasítás
`\begin{`**WHILE**`}[`*behúzás*`]{`*magasság*`}{`*feltétel*`}`	Ciklus
`\begin{`**IF**`}[`*arány*`]{`*magasság*`}{`*feltétel*`}`	Kétágú (igaz-hamis) elágazás
\ELSE	A hamis ág utasításainak kezdete
`\begin{`**CASE**`}[`*felt.mag.*`]{`*magasság*`}{`*ágak száma*`}`	Általános (többágú) elágazás
\WHEN`[`*relatív szélesség*`]{`*feltétel*`}` *utasítások*	A többágú elágazás egy ága
\WHEN*`[`*relatív szélesség*`]{`*feltétel*`}` *utasítások*	Pöcök nélküli elágazáság

Beállítási lehetőségek

Név	*Kategória*	*Alapérték*	*Leírás*
\stukiwidth	Hosszúság	10 cm	Alapértelmezett stukiszélesség
\stmheight	Hosszúság	18 pt	Magasságegység
\loopindent	Makró	`\stmheight`	Alapértelmezett ciklusmag-behúzás
\slashheight	Makró	0.666	Az elágazáspöckök magassága
\thicklines	Makró		Átkapcsolás vastag vonalakra
\thinlines	Makró		Visszakapcsolás vékony vonalakra

Példa

```
\begin{stuki*}[6cm]{Feltételes maxker}
  \stm{k,l := m-1, \hbox{hamis}}
  \begin{WHILE}{5}{\stm{k \ne n}}
    \begin{CASE}{3}{16}
      \WHEN[3]{\stm{\lnot \beta(k+1)}}
        \stm*{SKIP}
      \WHEN[5]{\stm{\lnot l \land \beta(k+1)}}
        \stm[3]{l,i,max:=\\\hbox{hamis},k+1,\\f(k+1)}
      \WHEN[8]{\stm{l \land \beta(k+1)}}
        \begin{IF}[35]{2}{\stm{f(k+1) \le max}}
          \stm*{SKIP}
        \ELSE
          \stm[2]{i,max:=\\k+1,f(k+1)}
        \end{IF}
    \end{CASE}
    \stm{k := k+1}
  \end{WHILE}
\end{stuki*}
```