MAKRÓZÁS
(GYAKORLAT)

0 MI A MAKRÓ?

A makró egy tevékenység „mintája”, „sablonja”.

0.1 Egy kőkorszaki példa – Makró-assembler

Gyakori volt, hogy sokszor ismétlődő kóddarabokat kellett a programba beilleszteni. Például két 2-byte-os érték összeadása úgy, hogy a processzornak csak 1-byte-os összeadó művelete van. Nagyjából ez az ismétlődő kód:

```
LD   A,XX
LD   B,YY
ADD  A,B
ST   A,ZZ
LD   A,XX+1
LD   B,YY+1
ADC  A,B
ST   A,ZZ+1
```

Az ismétlődő kódban legfeljebb az `XX`, az `YY` és a `ZZ` cserélődik ki másra. Ez adta az ötletet, hogy legyen mód az assemblerben arra, hogy lehessen definiálni és felhasználni ilyen „paraméteres sablonokat”. Valahogy így:

```
ADD2 XX,YY,ZZ
```

Az assembler felismerve, hogy nem processzorutasításról van szó, és korábban definiálva lett az `ADD2` nevű makró, akkor ezen a helyen a fenti kóddal (pontosabban azzal, ami a makró törzsében szerepelt) helyettesíti.

0.2 Egy frissebb példa – Word-ös körlevél

Az Office-on belüli szövegszerkesztő lehetőséget biztosít arra, hogy egy ún. törzs-, valamint egy adatdokumentumot megszerkesztve „összefuttassuk” egyetlen dokumentummá, amelyben egy-egy –jól elkülönített– rész tartalmazza a körlevél egy-egy paraméterhez tartozó „elemét”. A törzsdokumentum egy sablon, amely „szabad” szöveget és az adatdokumentum által konkretizált paramétereket tartalmaz a szöveg tetszőleges pontján, tetszőleges számúszor.

0.3. Egy Turbo Pascal-os példa – feltételes fordítás

A TP-ben mód van arra, hogy bizonyos feltételek teljesüléséhez rendeljünk egy-egy programdarabot, amelyet figyelembe vesz a fordításnál. A feltétel igen egyszerű: egy azonosító definiáltsága. Erre szolgál az alábbi direktíva:

```
{$DEFINE név}
```

Az azonosító-függő kód elhelyezését így oldhatjuk meg:

```
{$IF név}
… kóddarab1 …
{$ENDIF név}
```

… és a nem definiáltság esetén érvényes kód:

```
{$IFNOT név}
… kóddarab2 …
{$ENDNOTIF név}
```

A `kóddarab1` a név definiáltsága esetén fordítódik a kódba, amíg a `kóddarab2` éppen ellenkezőleg.

0.4 A makrófunkció egy szövegtranszformáció

A makrózás mint tevékenység tehát egy leképezés két szöveg között. Az input makródefiníciókat, makróhívásokat és egyéb, „szabad” (makrómentes) szöveget tartalmaz. Az output makrótlanított szöveg, amelyben a makróhívások helyén az adott makró törzsében lévő szöveg található, természetesen az aktuális paraméterekkel behelyettesítve.

1. A MAKRÓ SZINTAXISA ÉS SZEMANTIKÁJA

1.1 Elvárások

1. a makróval kapcsolatos szövegrészek elkülönüljenek az egyéb szövegtől
2. a makródefiníció és -hívás egyértelműen összekapcsolható legyen
3. a makródefiníciókban az azonosító, a paraméterek és a törzs felismerhető legyen
4. a makró azonosítójául és formális paraméteréül (majdnem) tetszőleges név legyen adható; majdnem = a hosszára korlát adható
5. nem feltétlenül kell megengedni a makrótörzsben más makró hívását; saját hívása azonban tilos
6. a makró paramétereinek száma korlátozható
7. bármely azonosító legyen szó, azaz alfanumerikus karaktersorozat, amely speciális (pl. elválasztó) jeleket nem tartalmaz

1.2 A makródefiníció szintaxisa

Az alábbi egy lehetséges elképzelés csupán.

```
!!DEF!! makróazonosító #paraméterazonosító1 …↵
Makródefiníció-mentes szövegsorok↵
!!ENDDEF!!↵
```

Megjegyzések:

- Mint látható a makródefiníció sorai nem keveredik a „szabad” szöveggel, sőt a fej-, a törzs- és a lábrész önálló sorokat alkot.

- Speciális, az adott szintaktikai környezetben nem szerepelhető jelkombinációkkal érjük el az egyértelmű azonosíthatóság. („!!DEF!!”, „!!ENDDEF!!”, „ #”)
- Az azonosítók természetesen !!-jelpárt és #-jelet sem tartalmazhatnak, sőt nem lehet sem „DEF”, sem „ENDDEF” (ahogy hamarosan a hívás szintaxisából kiderül). Megengedhető, hogy aláhúzás-jelet tartalmazzanak. Hosszukra épeszű korlátozás adható.

1.3 A makróhívás szintaxisa

Az egységesség a könnyű alkalmazhatóság záloga. Ennek fényében „logikus” a makróhívás alábbi szintaxisa:

… !!makróazonosító!! paraméter$_1$ …

A „…” jelentése: a hívás tetszőleges szövegkörnyezetben megengedhető, azaz soron belül, akár többször is.

1.4 A makrókifejtés szemantikája

1. a makróhívást meg kell előzze a definíciója (vagy nem! És mi következik ebből?)
2. a mekródefiníciók egyértelműek, azaz nem lehet azonos névvel több makrót definiálni (vagy mégsem! És mi következik ebből?)
3. a sikeres makróhívás után a definiált törzs kerül a szöveg adott pontján behelyettesítésre, de a hívási paraméterekkel aktualizálva.

2 A MAKRÓPROCESSZOR

Adatok leírása:

```
MDefTab = MDef*
MDef = MNév × PDb × MTörzs
MNév = Szó
PDb = Egész
MTörzs = Szöveg
  [Típusinvariáns: paraméterek gyanánt ##i## szerepel,
                   ahol i a paraméterlistabeli sorszáma]
```

… és egy másik lehetséges változat:

```
MDef = MNév × PDb × MPar* × MTörzs
MPar = Karakter*
  [Típusinvariáns: p:MPar ⇒ p='#'+sz ∧ sz:Szó]
…
MTörzs = Szöveg
  [Típusinvariáns: paraméterek gyanánt az MPar-beliek
                   szerepelnek]
```

Konfliktusok:

Összefonódási – Input = Makródefiníciók × Makróhívások × Egyebek

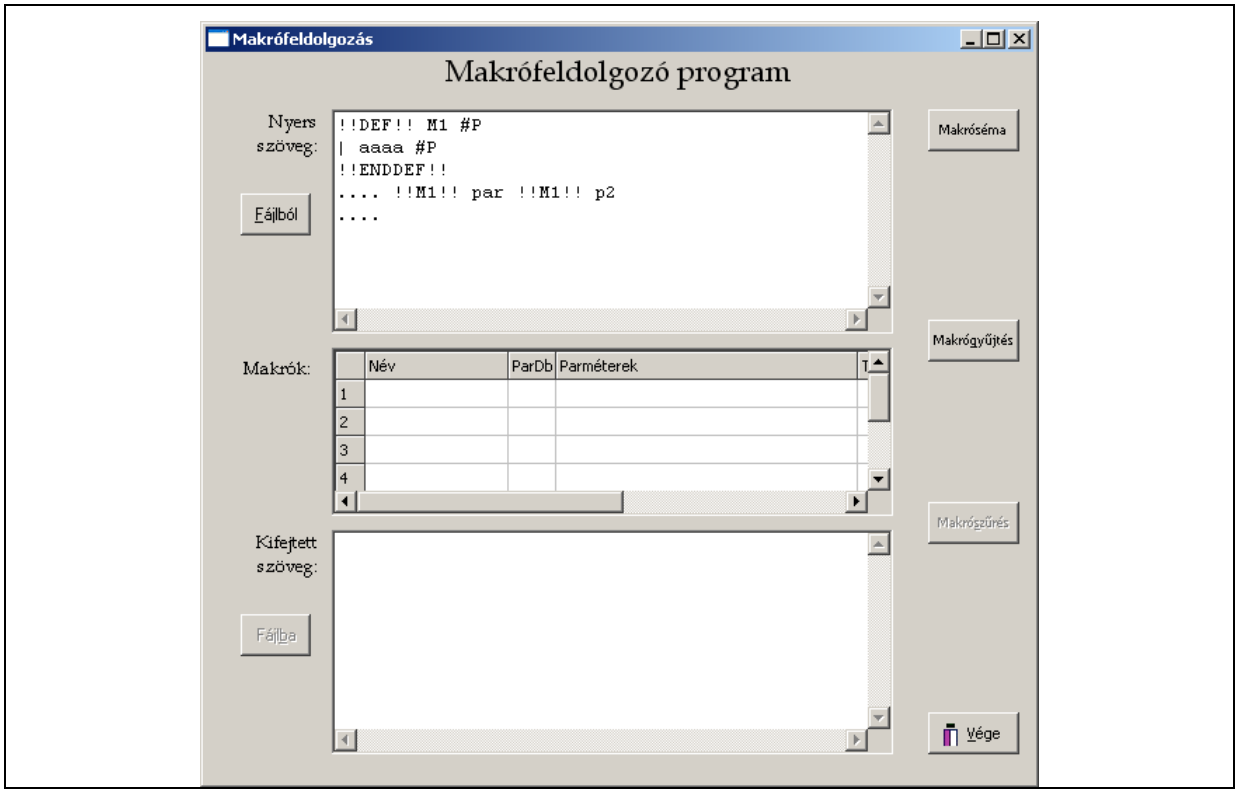
$$\Downarrow \qquad \Downarrow \qquad \Downarrow$$

$$\text{MDefTab} \quad \Rightarrow \quad \text{O} \qquad \text{O}$$

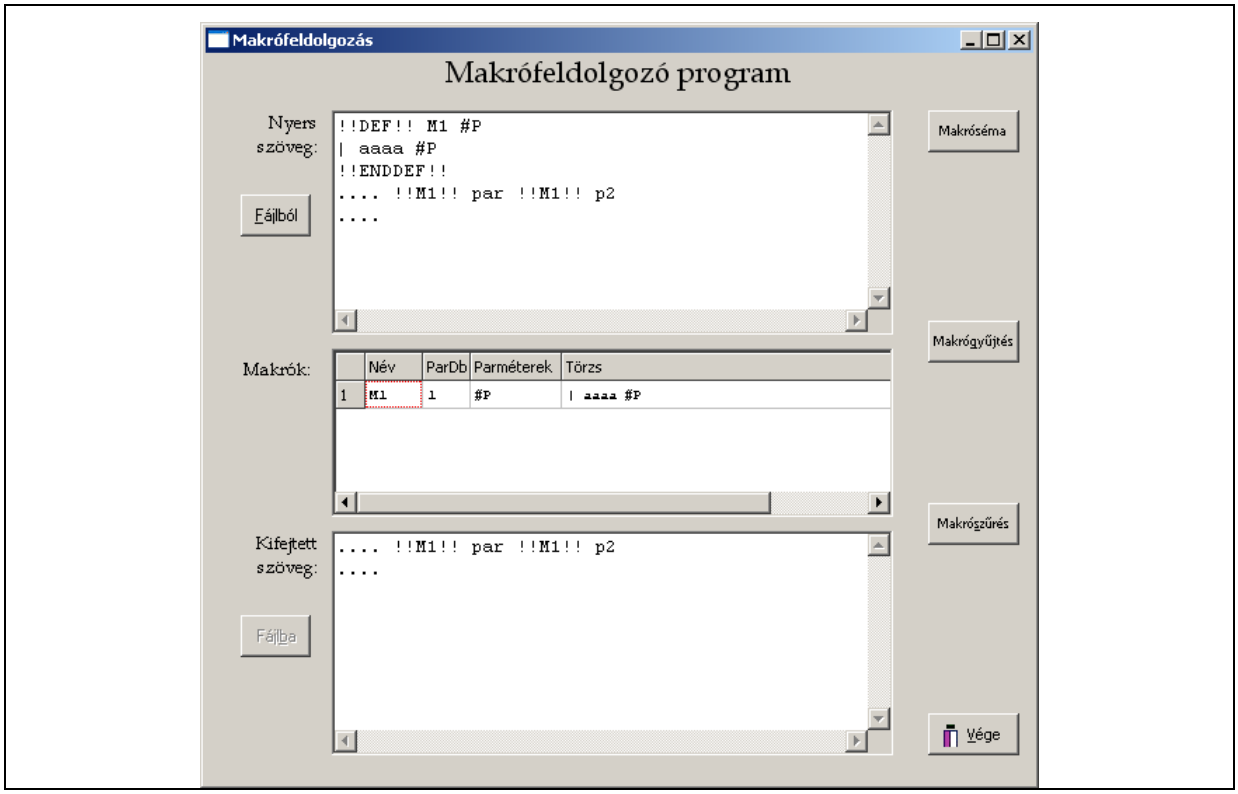
Tagolási – Input = Makródefiníciók / Makróhívások / Egyebek más-más tagolásúak

3 EGY MINTAALKALMAZÁS

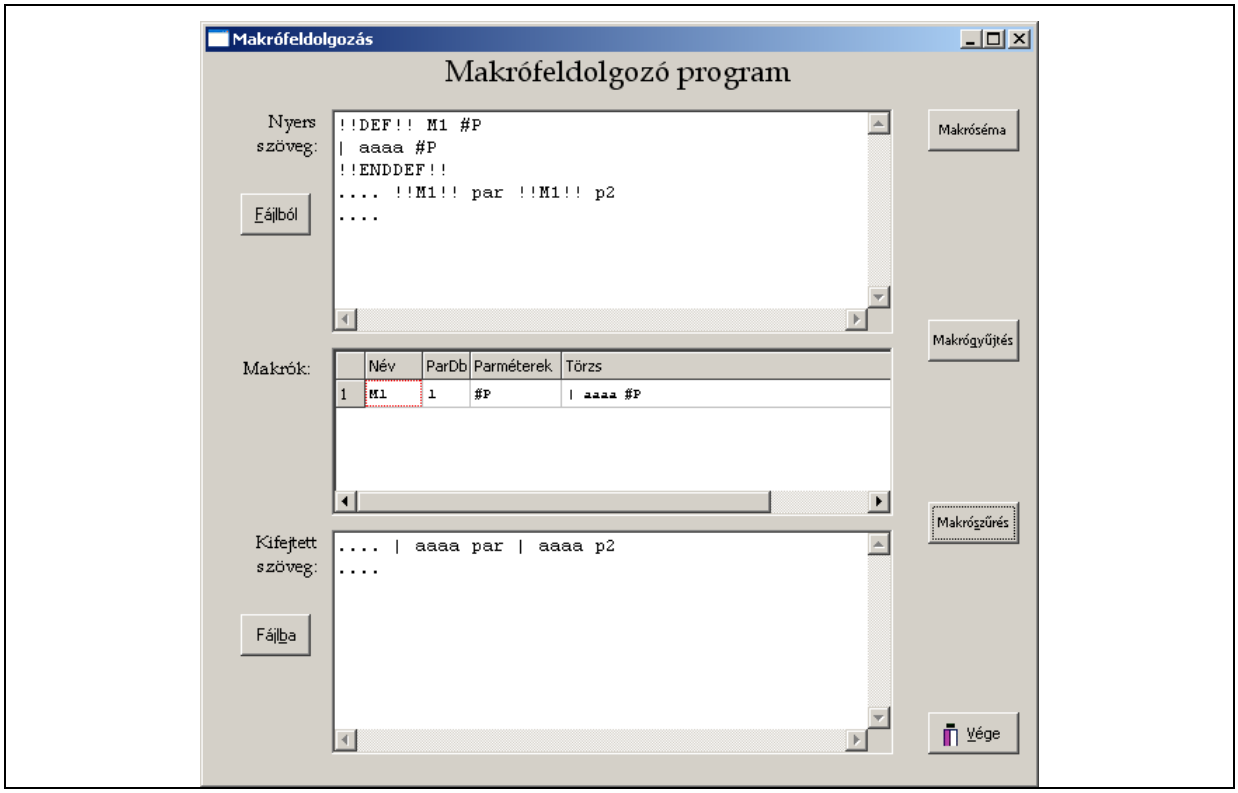
Alábbiakban néhány jellegzetes képpel mutatunk be egy lehetséges implementációt.



1. ábra. A mintaalkalmazás egy fázisa.



2. ábra. A mintaalkalmazás következő fázisa.



3. ábra. A mintaalkalmazás következő fázisa.

Próbálja ki: MakroFeldMO.exe! A keretprogramot letöltheti: makroGyak.zip.