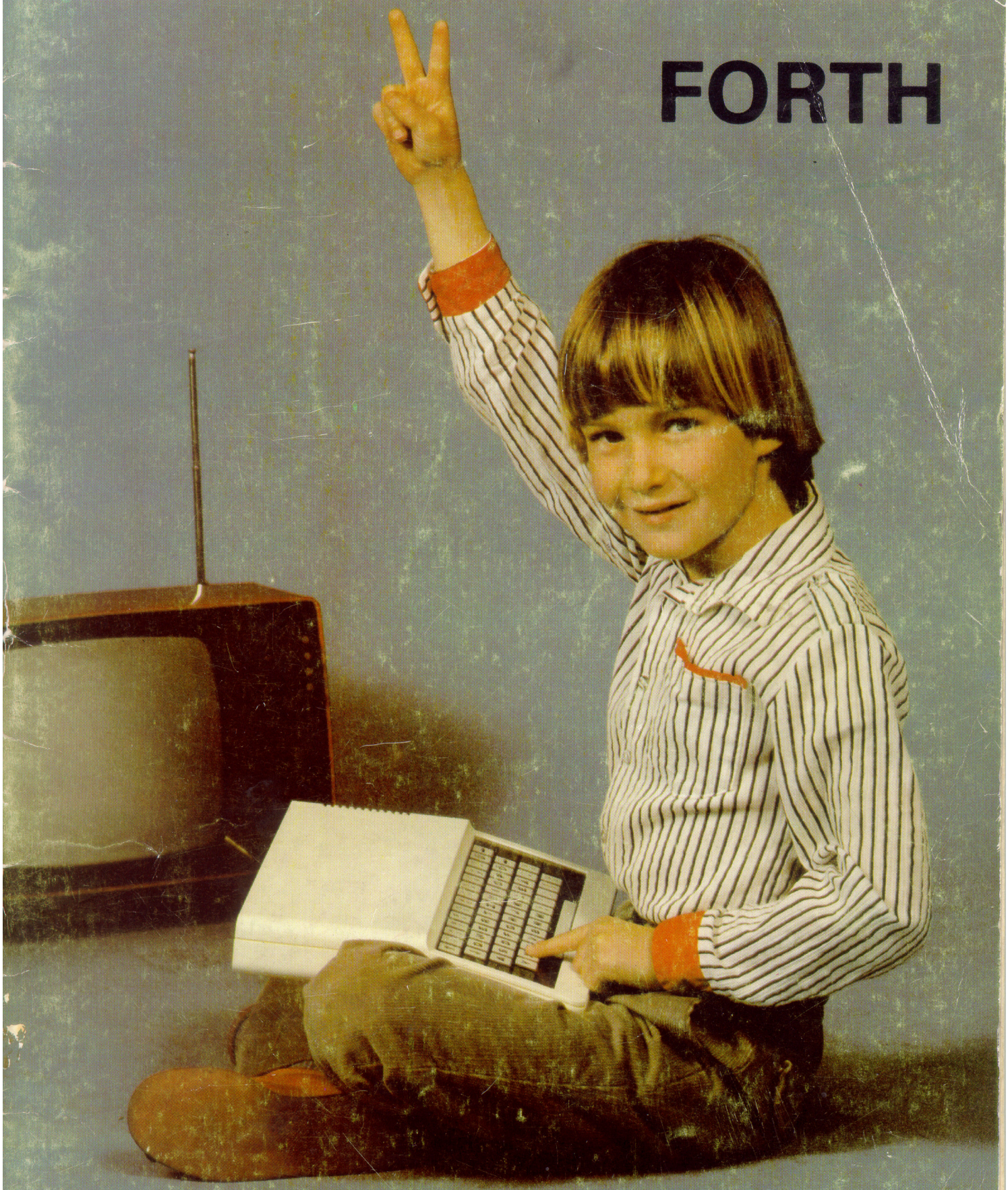
PRIMO FÜZETEK

FORTH



MICROKEY

COSY

PRIMO – FORTH

MTA–SZTAKI COSY MŰSZAKI FEJLESZTŐ LEÁNYVÁLLALAT
BUDAPEST 1985

Szerzők

MAROSI ISTVÁN
és
URBÁN ZOLTÁN

Lektor

ALEXIS GYÖRGY

Kiadja
az MTA–SZTAKI COSY Műszaki Fejlesztő Leányvállalat

Felelős kiadó

Móricz Sándor
igazgató

ISBN 963 311 203 6

Megjelent a GRAFO Kiadói Iroda GM gondozásában
2200 példányban, 10,8 (A/5) ív terjedelemben
185/85

VTV REPROTECHNIKA

85.016

A PRIMO FORTH BETÖLTÉSE

A FORTH rendszert a következő utasítással tölthetjük be kazettáról:

LOAD

az utasítás használatát a PRIMO felhasználói kézikönyvben megtalálhatjuk. A rendszer betöltődése után azonnal bejelentkezik, kiírva a rendelkezésre álló memória nagyságát is.

A minta szavakat tartalmazó bemutató program található ezután a szalagon. Ahhoz, hogy azt is betöltsük, gépeljük le a következő utasítást:

1 LOAD <RETURN>

(A "<RETURN>" a RETURN nevű billentyű lenyomását jelenti.) A számítógép a következő üzenettel válaszol:

SCR 1 olvasása. OK?

Ha most RETURN-t ütünk, működni kezd a kazetta betöltő rutin. Indítsuk el a szalagot, és nyomjunk RETURN-t. Az egyes tárolási egység (továbbiakban screen, "szkrín") betöltése néhány másodpercet vesz igénybe. A program megtalálását * karakterrel jelzi a gép.

Miután a screen betöltődött, a

SCR 2 olvasása. OK?

üzenet íródik ki, jelezvén, hogy ismét <RETURN>-t kell ütnünk. Ily módon összesen 7 db. screent kell beolvastatni, csak ezután állítsuk le a magnót.

1. ALAPUTASITASOK

A FORTH nyelv elsajátításának legkönnyebb módja a gyakorlás. Minthogy a FORTH párbeszédes (interaktiv) nyelv, le lehet ülni elé és kisérletezni. Ebben a kezelési útmutatóban sok mintapéldát találhatunk, amelyek szemléltetik a FORTH nyelv képességeit. Ajánljuk, hogy Ön is próbálja ki ezeket a saját számitógépén.

1.1. PARANCSBEVITEL

Betöltés után a FORTH rendszer bejelentkezik, és közli, hogy mennyi szabad memóriahely áll rendelkezésre. A villogó _ karakter (az ún. kurzor) mindig meg fog jelenni, amikor a gép vár valamilyen információt a billentyüzetről. Ekkor be lehet gépelni a RETURN-nel lezárt parancs sorokat. A legegyszerübb utasitás, amit kiadhatunk, az üres sor. Ha most RETURN-t nyomunk, a FORTH OK-val válaszol, minthogy látta, nincs semmi tennivalója, befejezte az aktuális sor feldolgozását, és új parancsra vár. Ezt kipróbálhatjuk néhányszor, hogy lássuk, FORTH-unk él, jól érzi magát, és figyel ránk.

Mivel a leirt parancsokat a FORTH először egy ideiglenes tárolóba (bemeneti puffer) helyezi, mindaddig, amig RETURN-t nem

ütöttünk, lehetőségünk van arra, hogy megváltoztassuk az utasitást vagy kijavitsuk a gépelési hibákat. A <-- billentyű segitségével törölhetjük ki a nem kivánt karaktereket (minden lenyomás egy jelet töröl a kurzor előtt), majd újra begépelhetjük a sor hátralévő részét. A bemeneti puffer sosem törlődik, hogy a korábban begépelt információt - vagy annak egy részét - újra fölhasználhassuk, gépelési munkát takaritva meg. A --> billentyű lenyomásával egy korábbi karaktert hivhatunk elő. Pl.: tegyük fel, hogy a kiadott parancs

1 4 + . <RETURN>

volt, és a gép válaszából kiderül, hogy valójában 2 4 + . -ot akartunk irni. Ekkor elegendő a 2 billentyűt megnyomni, majd 6 db. --> hatására a képernyőn megjelenik a 2 4 + . sor, amit RETURN-nel érvényesithetünk. Lehetőség van továbbá a pufferbe karaktereket beszúrni (SHIFT -->), és onnan karaktereket törölni (SHIFT <--). Pl. most a

SHIFT --> 1 --> --> --> --> --> --> -->

sorozat hatására a 12 4 + . parancsot, majd

SHIFT <-- SHIFT <-- 3 --> --> --> - _ .

beirásával a 3 4 - . utasitást küldhetjük el RETURN-nel. (A _ jel a szóköz billentyűt jelzi.) Használhatjuk szövegbevitelkor még a CLS billentűt is, hatására törlődik az egész képernyő és újra kezdhetjük a sor gépelését is. (Természetesen továbbra is visszanyerhetjük a korábban beirtakat.)

Munka közben előfordulhat, hogy programunk végtelen ciklusba

kerül, vagy a rendszer más ok miatt "lemerevedik" (pl. kazettáról való betöltéskor). A RESET gomb hatására a FORTH egy ún. melegindítást hajt végre, azaz úgy kerül alapállapotba, hogy a programjaink nem vesznek el.

1.2. SZAVAK

A FORTH utasítások alapegységét szónak nevezik. Más programozási nyelvek megfelelő kifejezései: utasítás, parancs, kulcsszó, (al)program, szubrutin, eljárás, stb. Miután befejeztük egy sor begépelését és RETURN-nel lezártuk, a FORTH rendszer értelmezője (külső interpretere) veszi át a sort. Felbontja különálló szavakra, amelyeket az előfordulásuk sorrendjében fog végrehajtani. Minden ilyen szónak van NEVE, amellyel hivatkozhatunk rá, és van egy DEFINICIÓJA, amely megmondja, hogy mit csinál. Ahol a szó nevének megfelelő karaktersorozat megjelenik, az egyben hivatkozást jelent az adott szóra. Begépeléskor az egyes szavakat szóköz (" ") választja el egymástól.

Amikor az értelmező dolgozik a soron, elővesz a következő szót (amelyet vagy szóköz vagy a sor vége követ). Ezután végignézi az ismert szavakat tartalmazó ún. szótárt, hogy kikeresse, mi ennek a szónak a "jelentése". Ha sikeresen megtalálta, végrehajtja a definícióban rögzítetteket. Ha nincs a szótárban az adott jelsorozattal egyező nevű szó, a FORTH feltételezi, hogy egy számot gépeltünk be, és megkísérli annak értelmezni. Ha az

aktuális számrendszerben nem létezhet ilyen alakú szám, "Nem definiált szó" hibaüzenetet kapunk, ami azt jelenti, hogy nem tud az adott utasítással az értelmező mit kezdeni. Ekkor a rendszer új parancssort vár, amit a kurzor újra megjelenése is mutat.

A szótárban induláskor csak olyan szavak vannak, amelyeket a rendszer maga bocsát a felhasználó rendelkezésére. (Ezek felsorolását l. a Függelékben.) Természetesen a felhasználó ezt tetszőlegesen kibővítheti olyan új szavak létrehozásával (definiálásával), amelyekben a már létező szavak bármelyikét felhasználhatja. (Bővebben l. az 1.7. fejezetet.)

A szavak nevében bármilyen látható karakter szerepelhet, kivéve a szóközt. Egy név tetszőlegesen hosszú is lehet, de csak az első 31 karakter kerül tárolásra; tehát két különböző szó első 31 karaktere nem egyezhet meg. 31 karakter igen sok, így lehetőségünk van olyan neveket adni, amelyek utalnak az illető szó funkciójára. A szavak kis- és nagybetűvel egyaránt beírhatók.

1.3. SZAMOK

A számok tetszőleges számrendszerben kifejezhetők (2-től 36-ig). Bekapcsoláskor a rendszer tizes számrendszert vesz fel. Ezt azonban bármikor megváltoztathatjuk a DECIMAL ill. HEX szavak valamelyikével, vagy tetszőleges más számrendszert is előírhatunk. Ezután ebben a számrendszerben lesz értelmezve minden beírt szám,

és kiíráskor is ennek megfelelő formátumot kapunk. Általában javasolható, hogy válasszunk egy számrendszert, és ahhoz tartsuk magunkat végig a programozás során. Ezzel elkerülhetünk bizonyos hibákat, mint pl. hogy elfelejtjük, mi az aktuális számrendszer, és mást írunk be valójában, mint amit szeretnénk.

A számokat pozitív vagy negatív egészekként adhatjuk meg. A pozitív (előjeltelen) számok 0 → 65535-ig terjedhetnek, az előjeles egész számok -32768 → 32767-ig. Lehetőség van dupla pontosságú számok használatára is, amelyek előjeles egészek -2147483648 → 2147483647-ig. A dupla pontosságú számokat 32 biten ábrázolja a gép, és megadásukkor a szám részeként egy "."-ot kell elhelyezni. Pl.: "437" szimpla, ".437" vagy "43.7" dupla pontos számokat jelentenek.

Amennyiben olyan számértéket gépelünk be, amely az adott számtartományba nem fér be, Túlcsordulás! hibaüzenetet kapunk.

Minthogy a FORTH minden számot kettes számrendszerben tárol, a számrendszer alapszámának célszerű változtatásával konverziót hajthatunk végre az egyes számrendszerek között. Például tizes alapról 16-osba átváltáshoz gépeljük be a következőt:

DECIMAL 191 HEX . ⟨RETURN⟩

mire a gép válasza: BF OK (Vigyázzunk, mert továbbra is 16-os

számrendszerben maradunk!)

1.4. A PARAMETER STACK

A számítógépek programozása közben nem elég csak azt "megmondani" a gépnek, hogy mit csináljon (ez a program), hanem rendelkezésére kell bocsátani azon adatok halmazát is, amelyen a megfelelő műveleteket végre kell hajtania. Ezeket az adatokat paramétereknek is nevezzük. A FORTH szavak túlnyomó része a paramétereket az ún. paraméter stackben (stack=verem, "sztek") várja, és az eredményeket is oda helyezi.

A paraméter stack egy LIFO (1) típusú stack. Ez a tárolócelláknak egy olyan különleges elrendezése, ami "emlékezik" arra a sorrendre, amelyben az egyes értékeket beírták. A később beírt adatokhoz lehet mindig előbb hozzájutni. Képzeljük el, hogy egy íróasztalra az elintézendő iratokat úgy teszik le, hogy egyszerűen egymásra helyezik őket. Amelyik irat a legkésőbb jött, az lesz legfelül, és aki feldolgozza az aktákat, ezt veszi először kézbe. Ha az adminisztrátor éppen dolgozik egy anyagon, és beállítanak egy sürgős aktával, akkor a kézben lévő irat visszakerül a kupac tetejére. Ha egy még sürgősebb akta jön, akkor az előző megy a halomra, stb. Befejezve valamelyik irat földolgozását így módon biztosan a következő legsürgősebb aktát

---

(1)n Last In First Out, azaz ami utoljára bement, az jön ki elsőnek.

lehet elövenni, vagyis azt, amit legutoljára tettünk az iratkötes tetejére.

FORTH-ban egy számot úgy tehetünk a stack tetejére, hogy egyszerüen begépeljük a parancssor részeként az aktuális számrendszerben. A "." nevü FORTH szó pedig elveszi a stack tetején álló számot, és kiirja azt a képernyöre (szintén az aktuális számrendszerben).
Például gépeljük be a következö sort:

2 4 6 8 <RETURN>

A stackben most a következök vannak: 8
6
4
2

Ha begépeljük, hogy:
. <RETURN>
akkor a válasz: 8 OK lesz.

A stackben ezután a következök lesznek: 6
4
2

Most irjuk be:
. . . <RETURN> 6 4 2 OK
A stack kiürült. Ha mégis . <RETURN>-t ütünk, a gép . ? üres a

stack! hibaüzenetet ad.

A FORTH-ban a paraméter stacken kivül van még egy másik stack is, amit 'RETURN STACK'-nek (1) hivnak, és az értelmező használja visszatérési cimek tárolására. Minden hibaüzenet kiüriti mindkét stacket.

1.5. ARITMETIKAI UTASITASOK

A PRIMO FORTH-ba beépitett aritmetikai utasitásokat az 1. táblázatban foglaltuk össze. (Ezeket használjuk a leggyakrabban. További hasonló utasitásokat találhatunk még a Függelékben.)

Mivel ezek a szavak is a paraméter stackben várják az adatokat, használatuk az ún. forditott lengyel ábrázolásmód segitségével történik, vagyis a művelet operandusait a művelet elvégzése ELÖTT kell a stackbe helyezni.
Pl. két számot összeadni és az eredményt kiirni a következőképpen kell:

```
DECIMAL 8 31 + . <RETURN>     39 OK
```

---

(1)n Return=visszatérés ("ritörn")

Nézzük meg az egyes részek hatását:

8 A stackre teszi az 8-as értéket.

31 A stackre teszi az 31-es értéket.

+ Leveszi a stack két legfelső elemét, összeadja őket, majd az eredményt a stackbe teszi.

A stack mélysége összességében tehát 1-gyel csökkent!

. Elveszi a stack tetején levő értéket, majd kiírja: 39 OK.

Végeredményben a stack pontosan úgy maradt, mint a parancssor kiadása előtt volt.

A FORTH az összehasonlításokat (2. táblázat) az ún. pozitiv logika szerint kezeli. Az eredmény logikai értelmezése:

0 - hamis

nem 0 - igaz.

Összehasonlításnál (mint < > = stb.) jegyezzük meg, hogy amit szokás szerint a relációjel jobb oldalára irunk, azt kell a stackbe legfelülre tenni, mig a bal oldali paraméter felülről a második legyen. Pl: A B < azt vizsgálja, hogy A<B, eltávolitja mindkét értéket a stackről és csak a megfelelő logikai értéket hagyja ott.

1.6. STACK MÖVELETEK

A leggyakrabban végrehajtott utasitások egy másik csoportját alkotják a stack müveletek. A legfontosabbakat a 3. táblázatba gyüjtöttük össze. Ezeket általában arra használjuk, hogy mindig az

általunk éppen meskivánt sorrendet alakitsuk ki a stackben, anélkül, hogy a benne tárolt paraméterek bármelyikét elvesztenénk. Érdemes ezen szavak használatát alaposan begyakorolni, mert a későbbiekben igen jó szolgálatot tesznek.

Miközben dolgozunk, két alapvető szabályt mindig vegyünk figyelembe:

1. Minden, ami bekerült a stackbe, jöjjön is ki onnan.
2. Sose próbáljunk meg több értéket elvenni a stackről, mint amit rátettünk.

1.7. ÚJ SZAVAK DEFINIÁLÁSA

Mint említettük, a FORTH rendszer által "készen kinált" szavakon kivül a felhasználó is definiálhat új szavakat. Tegyük fel, hogy munkánk során gyakran van szükségünk arra, hogy egy szám köbét meghatározzuk. Igen egyszerű olyan szót létrehozni, amely ezt elvégzi.

: KÖBE DUP DUP * * . ; <RETURN> OK

Nézzük meg, mit csinálnak az egyes komponensek:

: Új szó beépitésének a kezdetét jelöli.

KÖBE Az újonnan definiálandó szó neve (amilyen néven a szótárba bekerül majd).

DUP DUP * * . Azok a (már meglévő) FORTH szavak, amelyek sorozatával lesz az új szó hatása azonos. (Tulajdonképpen azt irja le, hogy mit kell tenni a szó végrehajtásakor.)

; Befejezi a definiciót. Az ezt követő szavak újra végrehajtásra kerülnek.

Miután elkészitettük a fenti definiciót, bármikor kiszámithatjuk és kiirathatjuk egy szám köbét. Pl:

```
 4 KÖBE (RETURN) 64 OK
 3 KÖBE (RETURN) 27 OK
```

Mi történt volna, ha a KÖBE szót azelőtt használjuk, mielőtt a definiciójáť beségeltük? A FORTH KÖBE ? Nem definiált szó hibaüzenettel visszautasitotta volna. Szerencsére a kezdő programozó számára a FORTH az előre definiált szavak gazdag szótárával rendelkezik. Például a következő meglévő és igen gyakran használt szónak: 1+ (amely eggyel megnöveli a stack tetején lévő értéket) az alábbi definiciót kellene adnunk:

```
: 1+ 1 + ;
```

Ha most a fönti meghatározást újra begépeljük, az 1+ (Már létező név) figyelmeztetést kapjuk, amely jelzi, hogy ilyen név már található a szótárban. Ez nem feltétlenül baj (bármely korábbi szó definiciójáť megváltoztathatjuk), de általában igyekezzünk

elkerülni. Az értelmező mindig a legutolsó azonos nevű szót találja meg. Vigyázzunk viszont arra, hogy ha egy már létező szó megadásakor hivatkoztunk valamely szóra, akkor ez a hivatkozás nem változik meg attól, hogy a hivatkozottal azonos nevű új szót definiálunk. Pl. ha beirjuk, hogy

: 1+ 2 + ;

ettől az összes eddig definiált szó helyes marad, csak amit ezután irunk meg (és használjuk benne az 1+ szót) fog furcsán működni.

Előfordulhat, hogy egy definició nem fér ki egy sorba. Az a fontos csak, hogy a sorban minden szó teljes egészében be legyen gépelve, ekkor nyugodtan elküldhetjük a rész-definiciót RETURN-nel. A sor feldolgozása után a FORTH várni fogja a folytatást, amit az is jelez, hogy nem ir OK-t.

1.8. MÖKÖDÉSI MÓDOK

A FORTH szövegértelmezője két módban működhet: közvetlen végrehajtási és forditási módban.

Közvetlen végrehajtási módban a parancssor minden szavát igyekszik kikeresni a szótárból és azonnal végrehajta őket.

Ezzel szemben forditáskor a szavak túlnyomó többsége nem kerül végrehajtásra. Ezek helyett egy rájuk utaló hivatkozás lesz a szótárba beforditva. A : szó az értelmezőt forditási állapotba

helyezi, a ; hatására pedig visszaáll a közvetlen végrehajtási mód.

Azért, hogy jól megkülönböztethessük a közvetlen végrehajtást a fordítástól, próbáljuk ki a következő példákat:

905 . <RETURN> 905 OK

Mint látjuk, ez az utasítás azonnal végrehajtódott (a számítógép röstön válaszolt).

: IRD-KI 905 . ; <RETURN> OK

Látszólag nem történt semmi, valójában a szót az értelmező leforditotta. Ha ez után leirjuk, hogy IRD-KI, a leforditott szó végrehajtódik, és kiirja a kivánt eredményt:

IRD-KI <RETURN> 905 OK

Ahhoz, hogy gyorsan tanuljunk, be kell gyakorolni az alapvető FORTH szavakat, és bátran kiséreljünk meg olyan új definiciókat irni, amelyekkel bizonyos dolgokat egyszerűbben lehet megcsinálni. Fejlesszünk ki magunknak egy olyan jelölésrendszert, amellyel rögzithetjük vázlatszerűen mindazt, amit csináltunk, hogy ne essünk ugyanabba a hibába többször is.

FORGET: ezzel az utasitással törölhetünk a szótár végéből (a legutoljára megadott szavak közül) egy vagy több szót. Pl.: definiáljunk három szót a következőképpen:

: SZO1 ... ; <RETURN>

: SZO2 ... ; <RETURN>

: SZO3 ... ; <RETURN>

Ezután SZO3-at a következőképpen lehet kitörölni:

FORGET SZO3 <RETURN>

SZO2-t viszont önmagában nem törölhetjük! A törlés ugyanis mindig a megadott szóra és az összes, később definiált szóra vonatkozik! Tehát

FORGET SZO2 <RETURN>

hatására SZO2 és SZO3 törlődik, de megmarad SZO1.

Előfordul, hogy egy új szó megírása közben hibázunk (nem létező szóra hivatkozunk, nem zárunk le egy vezérlési struktúrát, stb.). A szót újra kívánjuk írni, de előtte szeretnénk kitörölni a hibás definiciót. A FORGET-et használva azonban hibaüzenetet kapunk, mintha az a szó nem létezne. VLIST-tel kiíratva a rendelkezésre álló szavakat, a kitörlendő szót inverz betűkkel kiírva találjuk meg. Ilyenkor adjuk ki a SMUDGE szót, és ezután már végre lehet hajtani a törlést. (A név pedig inverzből visszaváltozik normállá.)

FELADATOK

1. Mi a különbség a DUP + és 2 * között?
2. Mi a különbség a DUP * DUP * és a DUP DUP * * között?
3. Mi a különbség az OVER SWAP és a SWAP OVER között? ("szvor")
4. Mi az értelme az OVER OVER sorozatnak?
5. Mi az értelme a ROT ROT sorozatnak?

1. TABLAZAT

ARITMETIKAI MÖVELETEK

Szó	Leírása	Példa stackre előtte			Példa stackre utána		
				Tető			Tető
+	Összeadás	9	6	2		9	8
-	Kivonás	9	6	2		9	4
*	Szorzás (előjeles)	9	6	2		9	12
/	Osztás (előjeles)	9	6	2		9	3
1+	Eggyel növelés	9	6	2	9	6	3
2+	Kettővel növelés	9	6	2	9	6	4
ABS	Abszolút érték képzés	9	-6	-2	9	-6	2
MAX	A két legfölső érték nagyobbika	9	6	2		9	6
MIN	A két legfölső érték kisebbike	9	6	2		9	2
NEGATE	Előjelváltás (kettes komplemes)	9	6	2	9	6	-2
MOD	Maradékképzés (osztás maradéka)	9	6	2		9	0
*/	Második és harmadik érték szorzata osztva az elsővel	9	6	2			27
*/MOD	Ua. csak a maradékot is adja	9	6	2		27	0
+-	Legfölső érték előjelével szoroz	9	6	-2		9	-6
/MOD	Második érték osztva a legfelsővel maradékot és hányadost is ad	9	6	2	9	0	3
AND	Bitenkénti logikai ÉS művelet	9	6	3		9	2
OR	Bitenkénti logikai VAGY művelet	9	6	3		9	7
XOR	Bitenkénti KIZÁRÓ VAGY művelet	9	6	3		9	5

Egyéb, valamint dupla pontosságú és kevert műveleteket is találhatunk a Függelékben. Ez utóbbiak D, M vagy U betűvel kezdődnek.

2. TABLAZAT

ÖSSZEHASONLITÓ MÖVELETEK

Szó	Leirás	Elötte	Utána
<	Ha a második a stackben kisebb, mint az elsö, -1-et ad, egyébként 0-t.	9 6 2	9 0
>	Ha a második nagyobb, mit az elsö, -1-et ad, egyébként 0-t.	9 6 2	9 -1
=	Ha a két legfölsö érték egyenlö, -1-et ad, egyébként 0-t.	9 6 2	9 0
0=	Ha a legfölsö érték 0, -1-et ad, egyébként 0-t	9 6 2	9 6 0
0<	Ha a legfölsö érték negativ, -1-et ad, egyébként 0-t	9 6 2	9 6 0

3. TABLAZAT

STACK KEZELÖ UTASITASOK

Szó	Leirás	Elötte	Utána
.	Kiirja a legfölsö értéket	1 2 3	1 2
DROP	Eltávolitja a legfölsö értéket	3 2 1	3 2
DUP	Megismétli a legfölsö értéket	3 2 1	3 2 1 1
?DUP	Megismétli a legfölsö értéket,	3 2 1	3 2 1 1
	ha az nem 0 vagy	3 2 0	3 2 0
OVER	Megismétli a másodikat a stack tetején	3 2 1	3 2 1 2
ROT	Megforgatja a fölsö 3 elemet 4	3 2 1	4 2 1 3
-ROT	Visszaforgatja a fölsö 3 elemet 4	2 1 3	4 3 2 1
SWAP	Megcseréli a két fölsö elemet	3 2 1	3 1 2
.R	A második értéket jobbra igazitva irja a legfölsö elem által meghat. szélességü mezöbe	3 2 1	3
R	A return stack tetejét a paraméter stackbe másolja	(RS) 20 30 1 2 3	(RS) 20 30 1 2 3 30

A Függelékben találhatunk egyéb stack kezelö utasitásokat is.

2. ADATTERÜLETEK LEFOGLALASA

A FORTH rendszer lehetővé teszi számunkra, hogy a memória bizonyos részeit lefoglalhassuk konstansok, változók vagy tömbök számára.

2.1. KONSTANSOK

Ha bizonyos konstans értékeknek nevet kívánunk adni, amellyel később hivatkozhatunk rájuk, a CONSTANT szót használhatjuk. Leggyakrabban olyan adatokat szoktunk ily módon névvel ellátni, amelyeket gyakran használunk, és a nevüket egyszerűbb leírni, mint magukat a számokat újra elökeresni; vagy ha annak a konstansnak az értéke függ a gépi környezettől.

Például a 3600 CONSTANT SEC/H ⟨RETURN⟩ OK létrehoz egy új szót, aminek a neve SEC/H, és a 3600 értéket rendeli hozzá. Miután ezt a definiciót begépeltük, a SEC/H leírása éppen úgy 3600-at tesz a stack tetejére, mintha 3600-at irtunk volna. Pl.: 3 SEC/H * kiszámitja, hogy 3 órában hány másodperc van. Természetesen egy konstans értékét nemcsak közvetlenül adhatjuk meg, hanem a stack tetején az értéket más módon is elöállithatjuk. Pl: 60 DUP * CONSTANT SEC/H OK anélkül tölti be az órában levö másodpercek

számát a konstansba, hogy látnánk a konkrét értéket.

Figyeljünk arra, hogy miután egy konstanst definiáltunk, annak (bináris) értéke független lesz az aktuális számrendszertől.

2.2. VALTOZOK

A VARIABLE ("veriébl") FORTH szó egy olyan tárolóhelyet definiál, amelynek értéke a későbbiekben valószínüleg változni fog. A tárolóhely mérete két byte (16 bit). Tegyük fel, hogy játékprogramunkban az eredményt egy változóban kívánjuk tárolni. Ekkor irjuk be, hogy

```
0 VARIABLE EREDMENY <RETURN>  OK
↑
kezdeti érték
```

A változóra a későbbiekben a nevével hivatkozhatunk. Azt, hogy mit kívánunk tenni vele, az utána következő szó mutatja meg. Ha egy változó értékét a stackre kívánjuk tenni, az & (1) szót használjuk. Pl.: az eredményünket az EREDMENY & <RETURN> paranccsal tehetjük a stackre.

Amennyiben a változónk értékére vagyunk kiváncsiak, a "?" szót célszerü használni. Pl.: EREDMENY ? <RETURN> 0 OK

---

(1)n et-jel.

Ha a változó értékét módositani akarjuk, tegyük be elöször az új értéket a stackbe, irjuk le a változónk nevét, majd használjuk a "!" szót. Pl.: ha az eredmény értékét 100-ra kivánjuk állitani, gépeljük be: 100 EREDMENY ! <RETURN> OK

A "+!" szó az adott változóhoz hozzáad egy 16 bites értéket. Például ha az eredményünket százzal meg kivánjuk növelni: 100 EREDMENY +! <RETURN> OK

Megjegyzések:

1. Egy változó nevének leirása a változónak megfelelö 16 bites tárrekesz cimét helyezi el a stackben. Az "&" pontosabb definiciója szerint a stackben található értéket kicseréli az annak megfelelö cimröl elövett 16 bites számmal. A "!" a stack tetején levö értéket cimnek tekinti, és erre a cimre teszi le a stackben második számot, stt.
2. A "?" definiciója: : ? & . ;
3. A paraméter stack célszerü felhasználásával a munkaváltozók száma jelentösen lecsökkenthetö.

2.3. AZ AKTUALIS SZAMRENDSZER MEGVALTOZTATASA

Az aktuális számrendszer alapszámát a FORTH egy BASE ("béz", alap) nevü változóban tárolja. A HEX ill. DECIMAL szavak ennek az értékét módositják. Amennyiben a tizes vagy tizenhatos számrendszeren kivül más alappal kivánunk dolgozni, irjuk át

közvetlenül a BASE változó értékét. A számrendszer alapszáma 2 és 36 között bármi lehet. Például ha kettes alappal akarunk számolni: 2 BASE ! ⟨RETURN⟩ OK és minden további kiírás a kettes számrendszernek megfelelően fog történni. Ne feledjük, hogy az adatok megadása is kettes alappal történik, tehát ezután pl. négyes számrendszerbe: 100 BASE ! ⟨RETURN⟩ OK térhetünk át.

2.4. TÖMBÖK

A tömbök igen sok alkalmazásban játszanak fontos szerepet. Például 10 változó (T0, T1, T2, stb.) helyett célszerűbb egy T-n kezdődő, 10 egymást követő elemet tartalmazó elrendezést használni. Ez nemcsak az elemek rugalmasabb kezelését biztositja, hanem lényegesen jobb memória kihasználást is lehetővé tesz. Az egyes elemeket azután megfelelő címszámitás útján lehet elérni.

Tömbök számára az ALLOT szóval foglalhatunk helyet a szótárban. Pl.: a fenti T tömbre:

0 VARIABLE T 18 ALLOT ⟨RETURN⟩ OK

Az egyes részek hatása:

0 VARIABLE T egy kétbyte-os T nevű változónak foglal helyet.

18 18-at tesz a stackre.

ALLOT további 18 byte-ot foglal le a tömb elemei számára.

Az n-edik tömbelemet a következőképpen érhetjük el: tegyük

n-et a stack tetejére, majd: 2 * T + & <RETURN> OK

2.5. EGYEB MEMORIAMOVELETEK

A továbbiakban négy olyan szót ismertetünk, amelyek segitségével a memória tartalmát közvetlenül módosithatjuk:

1. CMOVE A stack tetején lévő érték által meghatározott számú byte-ot mozgat át az egyik memóriacimről (harmadik a stacken) egy másik cimre (második érték a stacken). A legkisebb cimü byte másolódik elöször, majd sorban a többi.

   Pl.: 16396 8000 64 CMOVE <RETURN> OK

   elvileg 64 byte-ot mozgat át a 16396-os cimről a 8000-es cimre. (Valójában ROM-ba nem tud irni.)

2. FILL Egy adott memóriacimtől (harmadik a stacken) adott számú (második a stacken) byte-ot (elsö a stacken) lerakol.

   Pl.: 8000 64 255 FILL <RETURN> OK

   64 db. hexa FF byte-ot tesz le a 8000-es cimtöl kezdödöen.

3. ERASE Egy memóriarész kinullázását végzi. Megegyezik a 0 FILL utasitással.

   Pl.: 8000 64 ERASE <RETURN> OK

   kinullázza a 8000-en kezdödö 64 byte hosszú memóriablokkot.

4. BLANKS Egy memóriaterületet szóköz karakterrel tölt fel. Megegyezik a BL FILL utasitással.

Pl.: 8000 64 BLANKS <RETURN> OK

64 darab szóköz karaktert (ASCII hexa 20) tesz le a 8000-es címtől kezdődően.

4. TABLAZAT

MEMORIAMÖVELETEK

Szó	Leirása	Előtte	Utána
!	A stack tetején levő címre eltárolja a stacken masodik értéket	3 15355	üres
&	A stack tetején található cim tartalmát a stackbe teszi	15355	3
?	Kiirja a stack tetején megadott cimü memóriarekesz tartalmát	15355	üres
+!	A stacken második értékkel megnöveli a stack tetején megadott cimü memóriarekesz tartalmát	12 15355	üres
C&	Egy byte-ot felhoz a stack tetején megadott memóriacimről	15355	15
C!	Eltárol egy byte-ot (második a stacken) a stack tetején megadott memóriacimre	254 15355	üres

CMOVE ("szimáv")
FILL lásd szöveg
ERASE ("iréz")
BLANKS ("blenksz")

Dupla pontossagú müveletek: lásd Függelék.

FELADATOK

1. Definiáljunk egy CSEREL szót úgy, hogy felcserélje két változó tartalmát, azaz

   ha A és B változóknak lettek definiálva, akkor az A B CSEREL utasitások eredményeképpen A korábbi értéke B-ben legyen, B értéke A-ban.

2. Definiáljunk egy ATVISZ szót oly módon, hogy egy tömböt másoljon át egy vele azonos méretű másik tömbbe.

3. INPUT ÉS OUTPUT

Ahhoz, hogy bármilyen célra használhassuk a számítógépet, szükséges adatokat bevinni a komputerbe (input) és az eredményeket is olvashatóan kell megkapnunk (output). A FORTH sokféle lehetőséget nyújt ezekre.

3.1. ADATOK BEVITELE

A FORTH-nak nincs szüksége igazi beviteli utasitásra (mint pl. INPUT a BASIC-ben), mivel szinte minden utasitás a paraméter stacken várja a paramétereit. Ezeket általában a parancs végrehajtása előtt tesszük a stackre. Példaként tegyük fel, hogy a 4X↑3-3X+2 kifejezés értékét kivánjuk meghatározni valamely X-re. Ehhez célszerű alkotni egy olyan szót, amely egy szám köbét határozza meg, és definiálni egy másikat, amely a kifejezés többi részét számolja. Egy lehetséges megoldás:

```
: KÖB DUP DUP * * ; <RETURN>  OK
: KIF DUP KÖB 4 * SWAP <RETURN>
   3 * - 2 + . ; <RETURN>  OK
```

Most tehát van egy olyan szavunk (KIF), amely a stack tetején levő X értékkel kiszámitja a kifejezés értékét. Ehhez tegyük X-et a stackre a szó végrehajtása előtt a következőképpen:

```
8 KIF <RETURN>  2026 OK
```

Hasonló módon akárhány paraméter átadható, és nincs szükség beviteli utasitásra.

A FORTH rendelkezik egy olyan utasitással, amelyik egy karaktert képes beolvasni a billentyüzetről. Ez a szó a KEY. Hasonlóan használható, mint a BASIC INKEY$, azzal a különbséggel, hogy a KEY mindig megvár egy billentyü lenyomást, mig az INKEY$ nem. (A BASIC INKEY$-nak valójában a FORTH ?KEY szó fel meg, ez 0-át ad vissza, ha éppen nincs lenyomva egy billentyü sem.)

Tegyük fel, hogy futás közben szükségünk van valamilyen információra, amit a billentyüzetről várunk.
Pl.: 'KER HASZNALATI UTASITAST? (I/N)'
A KEY ezt követően a paraméter stackre teszi a leütött karakter ASCII ("eszki") kódját, amely vizsgálható.

Az előbb elmondottak ellenére a FORTH rendelkezik olyan szóval is, mellyel 1 darab több jegyü számot lehet beolvasni. Ez az INPUT nevü szó.

3.2. KIIRATAS

A FORTH többféle lehetőséget is kinál a kimenő információ megjelenitésére. A leggyakrabban használt kiiratási forma egy szövegsor. Az erre szolgáló FORTH szó a ." amelyet a kiirandó szöveg majd a záró " követ. Pl.:

." Ez egy sor szöveg" <RETURN>

az Ez egy sor szöveg üzenetet irja a képernyőre. Minden további kiiratás ugyanebbe a sorba fog kerülni. A CR FORTH szó szolgál arra, hogy új sort nyisson, és az ezt követő kivitel már az új sor elején fog kezdődni. A közvetlen kurzor pozicionálásra az AT szó szolgál. Pl.:

3 2 AT ." HELLO"

a képernyő harmadik sorának második oszlopába irja ki a fenti üzenetet. (Tehát sor oszlop sorrendben kell megadni a paramétereket!) Lekérdezhetjük azt is, hogy éppen most hol áll a kurzor, erre az AT? szó szolgál. Az előző paraméter-sorrenddel azonos módon adja vissza a paramétereket, a stack tetején tehát a vizszintes pozició lesz, vagyis, hogy melyik oszlopban vagyunk.

Amennyiben a képernyő utolsó pozicióját irunk, a FORTH automatikusan végrehajt egy "scroll"-t, azaz a képernyő minden sora eggyel feljebb kerül, a legfelső sor pedig elvész.

Egyetlen karaktert más módon az EMIT FORTH szó segitségével vihetünk a képernyőre. Ez az aktuális kurzorpozicióba kiirja azt a karaktert, amelynek ASCII kódja a stack tetején található. A megfelelő kódot legegyszerűbben a KEY paranccsal tehetjük oda. Az EMIT alkalmas arra is, hogy olyan karaktereket irassunk ki, amelyeket nem lehet a billentyűzetről elérni. Pl.:

142 EMIT <RETURN> @ OK a 142 ASCII kódú karaktert irja ki, amely a @ karakternek megfelelő kód.

3.3. SZAMOK KIIRATASA

Egy számot legegyszerűbben a jól ismert . FORTH szó segitségével irathatunk ki a képernyőre. Ez a stack tetejének tartalmát a lehető legkisebb szélességű mezőbe irja ki, azaz vezető nullák vagy szóközök nélkül. A kiirt számot egy szóköz követi. A következő FORTH szavak segitségével számok kiiratási formáját tetszőlegesen megváltoztathatjuk:

.R a számot egy megadott szélességű mezőben jobbra igazitva irja ki, szóközökkel kiegészitve a mező fennmaradó baloldali részét. Pl.:

22 4 .R <RETURN>

a 22-őt egy 4 karakter szélességű mező jobb szélére helyezve irja ki, azaz két vezető szóközzel kiegészitve.

<# képszerű (karakterenkénti) kiiratás szerkesztésének kezdetét jelzi, és egy dupla pontosságú számot vár a stacken. (1) A szám képének megszerkesztésére az alábbi FORTH szavakat alkalmazhatjuk:

hátulról előre haladva beteszi a szám következő jegyét egy pufferbe. (Természetesen ez is az aktuális számrendszerben értendő.) Pl.: ha a szám értéke 123, akkor az első # 3-at, a következő #-ok pedig sorban a 2,1,0,0,... karaktereket teszik a pufferbe.

#S a szám maradék jegyeit a pufferbe teszi. Ha már nincs több

---

(1)n Mivel általában egyszeres pontosságú számokkal szoktunk dolgozni, jó hasznát vesszük ilyenkor az S->D FORTH szónak, amely a stack tetején lévő 16 bites számot 32 bitessé konvertálja.

jegye, egy 0-t akkor is letesz.

HOLD a stack tetején vár egy ASCII kódot, és az ennek megfelelő karaktert teszi a puffer következő helyére. Pl.:

46 HOLD

egy pontot tesz a pufferbe.

#> befejezi a képszerű kiíratás szerkesztését. A stackben a puffer címét és hosszát adja vissza; így módon lehetővé teszi, hogy a kiíratást közvetlenül végrehajtassuk egy TYPE paranccsal.

Megjegyzés: a # és a #S a dupla pontos számot előjeltelen 32 bites értékként kezeli.

Példaként definiáljunk egy FIL nevű szót, amely egy egyszeres pontosságú, előjeltelen számot vár a stacken, azt fillérekben kifejezett értéknek tekinti, és kiírja

x,y FT

alakban, ahol x az egész forintokat, y pedig a maradék filléreket jelenti.

```
: FIL 0 <# # # # 44 HOLD #S #> <RETURN>
  TYPE ." FT" ; <RETURN> OK
```

Néhány alkalmazási példa:

```
   1234 FIL <RETURN> 12,34 FT  OK
   10000 FIL <RETURN> 100,00 FT  OK
   0 FIL <RETURN> 0,00 FT  OK
   -1 FIL <RETURN> 655,35 FT  OK
```

Mint már említettük, a képszerűen szerkesztett kiíratás

alkalmazásával tetszőleges alakban jelenithetünk meg számokat. Ehhez azonban megfelelő gyakorlásra szükség van.

3.4. EGYÉB KIIRATASI UTASITASOK

A következő FORTH szavak szintén a képernyő szerkesztését segitik:

SPACE ("szpész") egy szóköz karaktert ir az aktuális pozicióba. Megegyezik a 32 EMIT utasitással.

SPACES ("szpésziz") a stack tetején levő érték által meghatározott számú szóköz karaktert ir ki.
Pl.: 5 SPACES <RETURN> 5 szóközt ir a képernyőre.

CLS letörli a képernyőt és a kurzort a bal felső sarokba állitja. A következő kiiratás tehát fentről fog indulni.

3.5. GRAFIKAI LEHETÖSÉGEK

A FORTH megengedi, hogy a PRIMO adta grafikai lehetőségeket teljesen kihasználjuk. Beállithatjuk, hogy a rajzolás milyen "szinnel" történjék: fehérrel, vagy feketével. Az ON FORTH szó fehér szint, mig az OFF feketét állit be. Hatásuk addig érvényes, amig az ellenkezőt be nem állitjuk.

A PLOT szóval tudunk egy (fekete vagy fehér) pöttyöt kiirni a képernyőre:

Pl. 16 3 PLOT <RETURN> OK a grafikus képernyő alulról vett negyedik sorának tizenhetedik pozíciójába ír. (A sorrend tehát X Y PLOT.) Az X=0 Y=0 pont a képernyő bal alsó sarkában van. Egyenest a DRAW ("dró") szóval húzhatunk: az egyenes kiindulópontja az a pont lesz, amit utoljára beállítottunk (pl. a PLOT, vagy egy előző DRAW segítségével), végpontját pedig X Y sorrendben kell a DRAW előtt megadni. Pl. 30 20 DRAW <RETURN> OK esetünkben a (16,3) ponttól kezdve a (30,20) pontig húzott egyenest. Ugyanezt az egyenest meghúzhattuk volna a DRAWREL szóval is, ekkor a végpont koordinátái helyett a relatív koordinátákat kell megadni: ennyivel kell megnövelni az aktuális koordináta értékeket, hogy a végpontba jussunk. Az előző 30 20 DRAW helyett tehát írhattuk volna:

14 17 DRAWREL <RETURN> OK

hiszen 16+14=30, és 3+17=20. Az aktuális pont koordinátáit a FORTH az X és Y nevű változókban tárolja. Ezeket közvetlenül is átírhatjuk, de egyszerűbb, ha az új koordinátákat az XYSAVE ("exvájszév") szóval állítjuk be: ez nem rajzol semmit a képernyőre, de a következő DRAW vagy DRAWREL ettől a ponttól kezdve indul.

Egy pont állapotát az ON? szóval tudjuk lekérdezni. Például

52 0 ON? <RETURN> OK

A stackbe 0 kerül, ha a pont sötét, és -1 kerül, ha világos.

4. FELTÉTELES ELAGAZASOK ÉS CIKLUSSZERVEZÖ UTASITASOK

Mint láttuk, a FORTH értelmezöje egymás után veszi elö a begépelt szavakat, majd leforditja és eltárolja az új definiciókat, ill. végrehajtja az egyéb utasitásokat. A végrehajtási sorrend ekkor megegyezik a megadási sorrenddel. Nyilvánvaló, hogy egy használható programnyelvnek lehetöséget kell biztositania arra, hogy az utasitások logikai - végrehajtási - sorrendje eltérjen a fizikai - tárolási, begépelési - sorrendtöl. A hagyományos nyelvek (ASSEMBLER-ek, FORTRAN, BASIC, stb.) erre a célra feltétlen és feltételes ugró utasitásokat biztositottak, amelyekböl a felhasználó létrehozhatta a számára szükséges (program végrehajtási sorrendet) vezérlö struktúrákat. A FORTH azonban a programiró számára közvetlenül azokat a vezérlési formákat biztositja, amelyekre szüksége lehet. (1) Emiatt sohasem használunk explicit ugrásokat (egyébként is roppant körülményes lenne használatuk), hanem a megfelelö vezérlö utasitásokkal dolgozunk.

Fontos! A vezérlö struktúrák közvetlenül (egyszerüen begépelve) nem hajthatók végre: ezek kizárólag egy definició

---

(1)n A FORTH lényegéböl következik, hogy a felhasználó tetszöleges új vezérlö struktúrákat is szabadon definiálhat.

törzsében jelenhetnek meg.

4.1. FELTÉTELES ELAGAZÁSOK

A kétirányú feltételes elágazásokat három FORTH alapszóval vezérelhetjük: IF, ELSE és ENDIF (ez utóbbit THEN-nek is írhatjuk). Az IF egy logikai értéket vár a paraméter stack tetején, és attól függően ágaztatja el a végrehajtást. A feltételes elágazás formája:

```
: DEF felt. IF tev.1 ELSE tev.2 ENDIF tovább ;
```

ahol

: DEF — a definíció kezdete.

felt. — ez a rész egy logikai értéket (nulla/nem nulla) hagy a stacken.

IF — elveszi a stack tetején levő számot és megvizsgálja.

tev.1 — ezt a tevékenységet hajtja végre, ha a feltétel igaz, azaz a szám nem 0 volt.

ELSE

tev.2 — ez a rész kerül végrehajtásra, ha a feltétel hamis, azaz a szám 0 volt.

ENDIF

tovább — mindkét ág itt folytatódik.

Amennyiben az IF nem 0 értéket talál a stacken, minden végrehajtásra kerül az ELSE-ig ("elz"), majd a további szavak

kimaradnak, és az ENDIF után kerülnek csak újra végrehajtásra. Másrészről, ha a stacken az IF nullát talált, minden kimarad az ELSE-ig, és onnan folytatódik a programvégrehajtás. Az 'ELSE tev.2' részt el is lehet hagyni, ha nincs rá szükség.

A logikai értéket általában összehasonlító szavak (< > =) segitségével tesszük a stackre. Amennyiben a kivánt logikai értéknek épren az ellentettje áll rendelkezésünkre, a 0= alapszót használhatjuk. Ennek elvi definiciója:

```
: 0= 0 = ;
```

A FORTH lehetöséget biztosit 16 bites értékek között logikai müveletek elvégzésére. A megfelelö alapszavak: AND, OR és XOR. Ezeket a müveleteket a FORTH a két operandus között bitenként végzi el a 6. táblázatban található igazságtáblázatok szerint. Pl.:

```
1 1 AND  1-et ad   -1 1 XOR  -2-t ad   255 -256 OR  -1-et ad
```

A következö példánk egy olyan szó, amely a stack tetején talált értékről eldönti, hogy tizes számrendszerbeli számjegynek megfelelö ASCII kód-e (0-9). Ha nem, hibaüzenetet ad, ha igen, a neki megfelelö számértéket adja vissza:

```
: JEGY DUP DUP 47 > SWAP 58 < AND
  IF 48 - ELSE . ." Nem számjegy!" QUIT ENDIF ;
```

(A QUIT szót használjuk arra, hogy egy adott szó, valamint az öt hivó szavak futását félbeszakitsuk. BASIC megfelelöje a STOP.)

FELADATOK

1. Irjuk meg a <= >= és <> szavakat! (Kisebb vagy egyenlő, nagyobb vagy egyenlő ill. nem egyenlő).
2. Adjunk meg a "?DUP"-nak megfelelő definiciót -DUP néven! (Lásd 2. táblázat.)
3. Irjunk olyan szavakat MAXIMUM ill. MINIMUM néven, amelyek a stack tetején lévő két (16 bites) szám maximumát ill. minimumát adják.
4. Adjunk meg az ABS-nak megfelelő funkciójú szót ABSZOLUT néven!

4.2. FELTÉTELLEL VEZÉRELT CIKLUSOK

Amikor az utasitások egy csoportját többször ismételve kivánjuk végrehajtani, ciklust szervezünk. A ciklus elhagyása vagy valamilyen feltételtől függ (feltétellel vezérelt ciklus), vagy a végrehajtott ismétlések számától (ciklusszámlálóval vezérelt ciklus). Ebben a részben az előbbi csoporttal foglalkozunk.

A ciklusvezérlő utasitások egyik fajtája a kilépési feltételt a ciklus végén vizsgálja. Alakja:

: DEF BEGIN tev. felt. UNTIL tovább ;

ahol

: DEF a definició kezdete.

BEGIN megjelöli a ciklus kezdetét.

tev. az a tevékenység, amelyet ciklikusan kivánunk ismételtetni.

felt. a kilépési feltételt jelentő logikai érték előállitása.

UNTIL megvizsgálja a stack tetején lévő értéket, és ha az 0, visszalépteti a ciklus kezdetére (hamis feltétel).

tovább igaz feltétel (nem 0 érték a stacken) esetén a végrehajtás itt folytatódik.

Az UNTIL helyett END-et is irhatunk.

A következő példában egy olyan szót definiálunk, amely egy adott 16 bites szót keres a memóriában, és az első előfordulásának cimét adja vissza. Hiváskor a stack tetején az inditó cim, alatta a keresendő érték van:

```
: KERES 1- BEGIN 1+ OVER OVER @ =
  UNTIL SWAP DROP ;
```

a ciklus elején megnöveljük az aktuális cimet 2-vel, majd duplikáljuk a stack két legfelső értékét. Az adatfelhozatal után a keresett és a talált érték összehasonlitása következik: ha a két érték nem egyenlő, visszatérünk a ciklus elejére, egyébként eldobjuk a most már fölösleges keresett értéket és visszatérünk az aktuális cimmel.

A ciklus elhagyását irányitó feltétel a ciklus elején is állhat. Ekkor a ciklusutasitás a következő formát ölti:

```
: DEF BEGIN felt. WHILE tev. REPEAT tovább ;
```

ahol

: DEF a definició kezdete.

BEGIN megjelöli a ciklus kezdetét.

felt. a kilépési feltételt jelentő logikai érték előállitása.

WHILE ("vájl") megvizsgálja a stack tetején lévő értéket. Ha az 0, akkor a végrehajtás a "tovább" részen folytatódik, egyébként

tev. végrehajtja azt a tevékenységet, amelyet ciklikusan kivánunk ismételtetni.

REPEAT ("ripit") visszaugrik a kilépési feltétel előállitásához (BEGIN)

tovább a ciklusból kilépve ide kerülünk.

Figyeljük meg, hogy BEGIN - UNTIL ciklusnál a "felt." a ciklusból való kilépés feltételét, BEGIN - WHILE - REPEAT esetén a ciklusban maradás feltételét jelenti!

Példaként irjunk egy szót, amely a stack tetején álló (pozitiv) számról eldönti, hogy hány decimális jegyből áll:

```
: DECJEGY 1 SWAP BEGIN 10 / ?DUP
  WHILE SWAP 1+ SWAP REPEAT ;
```

A számot mindaddig osztjuk tizzel, amig eredményül 0-t nem kapunk. A számláló 1-ről indul. Pl.: a 38 kezdeti értéknél az első lépésben a 38-ból 3 lesz. Mivel ez nem 0, végrehajtjuk a ciklusmagot (a számlálót megnöveljük), majd újra osztunk. Ekkor már 0-t kapunk, igy a ciklusból kilépve a számláló értékével visszatérhetünk.

Ebben a részben emlitjük meg a ciklusutasitások egy különleges fajtáját, a végtelen ciklust is. Alakja:

: DEF BEGIN tev. AGAIN ;

("ögéjn"), amely a "tev." tevékenységet ismétli vég nélkül. Ezt a fajta ciklust elhagyni csak különleges módokon lehet (hiba, QUIT, külső beavatkozás, stb.).

FELADAT

Írjuk meg a BEGIN - UNTIL utasításhoz megadott példát BEGIN - WHILE - REPEAT felhasználásával!

4.3. CIKLUSSZÁMLÁLÓVAL VEZÉRELT CIKLUSOK

Ez a fajta ciklus nagyon hasonlít a BASIC FOR-ciklusára: adva van egy ciklusszámláló, amely egy meghatározott kezdőértékről indul, és mindaddig változik az értéke egy definiált lépésközzel, amíg az adott végértéket el nem éri. Az egyik formája a következő:

: DEF vég kezd. DO tev. LOOP ;

ahol

: DEF definiáljuk a DEF nevű szót.

vég a végértéket teszi a stackbe.

kezd. a kezdőértéket teszi a stackbe.

DO ("du") a ciklus kezdetét jelöli.

tev. a ciklikusan végrehajtandó tevékenység.

LOOP ("lúp") megnöveli a ciklusváltozó értékét eggyel, és megnézi, hogy elérte-e a végértéket. Ha igen, befejezi a ciklust, egyébként folytatja az új ciklusértékkel.

Jól figyeljük meg, hogy FORTH-ban a ciklus a végértékkel már nem fut le!

A cikluson belül a ciklusváltozó értéke lekérdezhető: az I szó a ciklusváltozó aktuális értékét teszi a stackre. Például egytől 10-ig írja ki a számokat a következő szó, mindegyiket új sor elejére:

```
: 1-10 11 1 DO CR I . LOOP ;
```

Mód van arra is, hogy a lépésköz ne 1 legyen. Ekkor a következő formát használhatjuk:

```
: DEF vég kezd. DO tev. lép. +LOOP ;
```

ahol

: DEF — a definíció kezdete.

vég — a végértéket teszi a stackbe.

kezd. — a kezdőértéket teszi a stackbe.

DO — a ciklus kezdetét jelöli.

tev. — a ciklikusan végrehajtandó tevékenység.

lép. — a stackre teszi a kívánt lépésközt.

+LOOP — megnöveli a ciklusváltozó értékét a stacken lévő számmal, és megnézi, hogy elérte-e a végértéket. Ha igen, befejezi a ciklust, egyébként folytatja az új ciklusértékkel.

A lépésköz lehet negatív szám is, ekkor a ciklusváltozó értéke folyamatosan csökken, amíg kisebb nem lesz a végértéknél.

Amennyiben a ciklust el kívánjuk hagyni, mielőtt a ciklusváltozó a végértéket elérte volna, a LEAVE ("lív") szót használhatjuk.

4.4. VEZÉRLÉSI STRUKTÚRÁK EGYMÁSBAÁGYAZÁSA

A vezérlő struktúrák "tev." tevékenység részén belül újabb vezérlési formák is elhelyezhetők. Az egyetlen megkötés, hogy az egymásbaágyazások megfelelőek legyenek, azaz az egyes struktúrák nem keresztezhetik egymást!

Például:

helyes:

```
... IF 10 0 DO ... LOOP ... ENDIF ...
```

helytelen:

```
... IF 10 0 DO ... ENDIF ... LOOP ...
```

A fenti szabály alól egyetlen kivétel van: a LEAVE szó beágyazható egy feltételes utasításba, azaz

```
... 10 0 DO ... IF LEAVE ELSE ... ENDIF LOOP ...
```

helyes.

Természetesen DO - LOOP ciklusokat is egymásba ágyazhatunk. Az előzőleg megismert I szó ebben az esetben mindig az aktuálisan legbelső ciklus ciklusváltozójának értékét adja vissza. A következő szint ciklusváltozóját is elérhetjük azonban a J szóval, sőt a K a még eggyel magasabb szint ciklusváltozóját teszi a stackbe.

Példaként tekintsük a következő programot, amely egy szorzótáblát ír a képernyőre:

```
: SZORZOTABLA CR CR 4 SPACES
     10 1 DO I 3 .R LOOP CR CR
     10 1 DO I 3 .R SPACE
       10 1 DO I J * 3 .R LOOP
     CR LOOP CR ;
```

Jól figyeljük meg, hogy ugyanazt a ciklusváltozót egyszer I-vel, egyszer J-vel értük el a beágyazódás mélységétől függően!

FELADATOK

1. Definiáljunk egy HATVANY nevű szót úgy, hogy "m n HATVANY" m-nek az n-edik hatványát számitsa ki. Ha n negativ, adjunk hibaüzenetet!
2. Irjuk meg a FAKTOR szót, amely a stacken lévő szám faktoriálisát számitja ki!

6. TABLAZAT

AND	1	0
1	1	0
0	0	0

OR	1	0
1	1	1
0	1	0

XOR	1	0
1	0	1
0	1	0

5. SZÖVEGSZERKESZTÉS ÉS -TÁROLÁS

Megismertük az előző részekben, hogyan lehet egy új szó definícióját közvetlenül begépelni. Ennek a módszernek a következő hátrányai vannak:

- hibás szavak javítása igen nehézkes és a teljes szöveg újbóli begépelésével jár;
- a begépelt szöveg elvész, azaz a képernyő letörlése után a szó definíciója (ebben a formában) nem nyerhető többé vissza;
- nincs lehetőségünk a beírt programokat pl. magnókazettára elmenteni, úgyszintén
- mások által megírt programokat nem tudunk átvenni.

Ezért fontos lehetőség az, hogy a begépelni kívánt programjainkat úgy is tárolhassuk a tárban, mint ahogy papírra leírnánk őket. Amikor a programunk készül, erre a "papírra" írjuk az utasításokat, definíciókat; szükség esetén "radírozunk", beszúrunk, kiegészítünk. Amikor úgy érezzük, hogy készen vagyunk, vagy ki akarjuk próbálni a begépelteket, átadjuk a "papírlapot" a FORTH-nak, hogy "olvassa el", azaz tekintse úgy a rajta lévő információt, mintha éppen most gépelnénk be a klaviatúrán. Eközben azonban a "papírlapunkon" lévő információ nem változik meg, tehát hiba esetén újra elővehetjük, javíthatunk rajta, újra beolvastathatjuk. Munkánk végeztével a "papíron" lévő információt kazettán

tárolhatjuk.

A FORTH-ban a "papírlapot" screennek ("szkrín") nevezzük, és mérete kötött: 1 Kbyte. Ez logikailag 32 db. 32 karakteres sort jelent. Az első sorra a 0-ás, az utolsóra a 31-es sorszámmal hivatkozhatunk. A számítógép memóriájában csak néhány screennek van hely, azokat a "lapokat", amelyek nem fértek be, magnókazettán lehet tárolni. Minden screennek sorszáma van, ez 1-től 255-ig terjedhet. Ha kíváncsiak vagyunk arra, hogy gépünk memóriájában hány screen lehet egyszerre, gépeljük be: #BUF . <RETURN>

5.1. SZÖVEGSZERKESZTÉS (EDITÁLÁS)

Szövegszerkesztésnek azt a tevékenységet nevezzük, ami közben "papírlapunkra" írunk, vagy rajta változtatást (törlést, beszúrást) hajtunk végre. Minden szövegszerkesztést az EDITOR parancs kiadásával kezdünk, és a FORTH szóval fejezünk be. Szövegszerkesztési üzemmódban minden parancssorra várást az E> karakterekkel jelez a gép.

Az editáláshoz (1) kétféleképpen kezdhetünk hozzá: vagy "tiszta lappal" kívánunk indulni, vagy egy (akár mágnószalagon) meglévő screenen akarunk változtatni. Az előbbi esetben az EDITOR parancs után az

5 CLEAR

---

(1)n Ez az angol edit (szerkeszt) szóból "magyarosított" kifejezés eléggé elterjedt számítástechnikai körökben: szövegszerkesztést jelent.

az utóbbiban az

5 LIST

utasitást adjuk ki. Az 5 helyett a stack tetején ezek a szavak természetesen bármilyen más screen-számot is találhatnak. Amennyiben a LIST az adott sorszámú screent nem találja meg a memóriában, a

SCR 5 olvasása. OK?

üzenet iródik ki, és a FORTH egy karakter leütésére vár. Ez után (ha a lenyomott billentyű nem a BRK) megpróbálja magnóról betölteni a kérdéses screent. Ha ez nem sikerül, hibaüzenetet kapunk. Sikeres betöltés esetén (vagy ha a screen már bent volt a memóriában) a screen a képernyőre listázódik. (1) A CLEAR ("klir") parancs sohasem keres kazettán, hanem új, üres screent nyit a memóriában, vagy ha ugyanilyen sorszámú screen ott létezik, azt irja felül szóközökkel.

Megtörténhet, hogy a számitógépben nincs hely a kijelölt screen tárolására. Ez vagy a CLEAR-nél fordulhat elö, vagy akkor, amikor a LIST-nek szalagról kell behoznia a screent. Ekkor két eset van: vagy van a memóriában olyan screen, amelyet nem változtattunk meg a legutóbbi beolvasása vagy mentése óta (azaz kazettán is megtalálható a pontos mása), ekkor ennek helyére kerül az új screen; vagy pedig egy bentlévö screen szalagra rögzitésére

---

(1)n A FORTH minden listázáskor számon tartja, hogy eddig hány sort irt a képernyőre. Ha a kiirt lista "el akar szaladni", azaz listázott sorok vesznének el, a FORTH megáll és vár egy karakter leütésére. Csak ezután hagyja a képernyöről elveszni az információt.

kerül sor.

A következőkben a szövegszerkesztéshez használt legfontosabb utasításokat ismertetjük; további hasznos szerkesztő szavakat találhatunk a Függelékben. Mindazok a szavak, amelyek egy sornak a számát igénylik, a stack tetején várják azt.

L az aktuálisan szerkesztett screent kilistázza a képernyőre.

P a screen egy sorát átírja az utasítást követő szöveggel, egészen a sor végéig. Pl.:

0 P Ez lesz a screen első sorában <RETURN>

17 P : KÖB DUP DUP * * ; 5 KÖB . <RETURN>

S a megadott számú sortól kezdve minden sort eggyel lejjebb mozgat, ez a sor üres lesz, míg az utolsó (eddigi 31-es) elvész. Pl.:

5 S <RETURN>

hatására a korábbi 5-ös sor a 6-os lesz, a 6-os a 7-es, sit, az új 5-ös pedig szóközökkel lesz tele. Ily módon lehet új sorokat beszúrni a régiek közé.

D a megadott számú sor kitörlése. (1) Minden ezt követő sor eggyel feljebb lép, a legutolsó pedig szóközökkel lesz tele. Az S-sel ellentétes művelet. Pl.:

12 D <RETURN>

hatására az új 12-es sor a régi 13-as lesz, az új 13-as a

---

(1)n A kitörölt sor nem vész el, hanem a PAD ("pad") nevű szövegpufferre kerül, ahonnan visszanyerhető (R, I, ld. Függelék).

régi 14-es, sit.

A az S-hez hasonlit, de az utasitást követö szöveget szúrja be az eddigi sorok közé. Pl.:

9 A Ez a 9-esbe, 9-es 10-esbe, sit. <RETURN>

5.2. SCREENEK ELMENTÉSE

Egy screen megszerkesztése után kérhetjük annak háttértárolóra történö mentését. Ezt a folyamatot a FORTH is kezdeményezi, ha nincs hely a memóriában egy új screennek. A magnókazettára történö kiirást ugyanúgy kell végrehajtani, mint pl. a BASIC programok mentését. A megfelelö FORTH utasitás a SAVE, amely a stacken talált sorszámú screent viszi szalagra. Pl.:

3 SAVE <RETURN>

elöször kiirja:

SCR 3 mentése. OK?

Ha a magnón megnyomtuk a RECORD és a PLAY gombokat, üssünk le egy RETURN-t, és a screen kivitele elindul. A BRK gombbal a folyamatot le lehet állitani.

A kivitel sikeres voltát a TEST szóval ellenörizhetjük: ugyanúgy használjuk, mint a LIST-et, de az adott sorszámú screen csak összevetésre kerül a memóriában lévövel. Eltérés esetén hibaüzenetet kapunk.

Ha kiváncsiak vagyunk arra, hogy egy adott pillanatban milyen screenek találhatók a memóriában, használjuk a SCREENS? szót,

amely a képernyőre irja a megfelelő sorszámokat.

5.3. SCREENEK BEOLVASTATASA

Amikor az elkészült "papirlapot" át kivánjuk adni a FORTH-nak, illetve kazettán tárolt programot akarunk betölteni, a LOAD ("lód") szót használjuk, a stackre téve a beolvasandó screen sorszámát. Amennyiben a kérdéses screen nincs még a memóriában, a LIST-tel egyező módon kerül be oda a szalagról. Mivel a screenen tárolt információt LOAD-nál a FORTH pontosan úgy kezeli, mintha az éppen akkor lenne begépelve, nemcsak definiciókat, hanem bármilyen egyéb utasitást is irhatunk a screenekbe. Van ezenkivül két szó, amelyet csak a "papirlapunkon" használunk:

--> ez megfelel az n 1+ LOAD utasitásnak, ahol n az aktuális screen sorszáma. Akkor használjuk, ha programunk nem fér el egy screenen. Ilyenkor a közbülső "lapokon" elhelyezzük ezt a "folytatás a következő lapon" jelet, és ebből a FORTH betöltéskor tudni fogja, hogy az eggyel nagyobb sorszámú screent is be kell töltenie.

(ezzel a szóval megjegyzést helyezhetünk el a screenben, azaz olyan szöveget, amelyet a FORTH LOAD-nál "lenyel", figyelmen kivül hagy. A megjegyzésben bármilyen karakter előfordulhat, kivéve a) csukó zárójelet, amely a kommentár befejezését jelenti. Célszerű minden screen első sorát megjegyzésnek fenntartani, amelybe beirjuk, hogy mi található ezen a screenen.

6. EGYÉB HASZNOS UTASITASOK

A FORTH alapszótára (azok a szavak, amelyeket a FORTH bekapcsoláskor "tud") igen gazdag, ezért itt most csak néhány hasznos szó kerül ismertetésre. A Függelék tartalmazza a felhasználói szavak teljes listáját és rövid ismertetésüket. Érdemes ezek közül minél többet kipróbálni, hogy az új szavak definiálásánál mind több feladatot tudjunk megoldani a lehető legcélravezetőbben.

BYE segitségével tudunk visszatérni BASIC-be.

VLIST ez az utasitás kilistázza az aktuális szótárat, tehát a rendelkezésünkre álló szavakat. Minden definició után ellenőrizhetjük vele, hogy az új szó is megjelent a szótárban.

TASK ez egy olyan szó, ami nem csinál semmit, de van. Ezzel kezdünk el általában minden új definició-sorozatot, és amikor törölni akarunk, tudjuk, hogy a TASK-tól kell kezdeni. Amennyiben a programunkat screenbe irtuk, és többször is javitunk rajta, egy új LOAD-nál figyelmeztető üzenetek sorát kapjuk, hiszen az előbbi próbálkozásnál már létrehoztuk ezeket a szavakat, és most minden szó (legalább) két példányban foglalja a memóriát. Célszerű ezért minden új programot a

következőképpen kezdeni:

FORGET TASK : TASK ;

IMMEDIATE ("imédiét") Mint emlitettük, forditási állapotban a szavak nem kerülnek végrehajtásra, hanem bekerülnek az új definicióba. Szükség van azonban olyan szavakra is, amelyek akkor is végrehajtódnak, ha forditási állapot van. Jó példa erre a ; szó, amely nem tudná másképpen kifejteni hatását, azaz a definició lezárását, hanem lefordulna. Ilyen szavakat mi is létrehozhatunk, ha a szó megadását követően leirjuk, hogy IMMEDIATE. Pl.:

: FORD ." Most forditási állapot van" ;

IMMEDIATE (RETURN)

Ezután a FORD szót alkalmazva egy definicióban, a fenti üzenet forditáskor fog megjelenni, az új szóba pedig a FORD nem kerül be.

LITERAL definició irása közben gyakran előfordul, hogy egy olyan állandót kell felhasználni, amely valamilyen számitás eredményeképp adódik. Ekkor megtehetjük persze, hogy magát a számitást tesszük bele az új szóba, pl.: a következő szó a stack tetejéhez (1+3)*2-t ad:

: 8+ 1 3 + 2 * + ;

A megoldás hátránya, hogy a számitást a gép a szó minden egyes végrehajtásakor újra elvégzi, jóllehet az eredmény mindig ugyanaz. Viszont az sem célszerű, hogy a konstans értékét mi magunk számitsuk ki. A problémát

a következő eljárással lehet megoldani: definició közben kapcsoljunk vissza közvetlen végrehajtási módba, végezzük el a kivánt számitást az eredményt a stacken hagyva, majd térjünk vissza a szó megadásához, és ott hivatkozzunk a stack tetején található értékre. A LITERAL szó az, amely definició közben hivatkozik a stack tetején lévő számra, a

((alapszó kapcsolja vissza időlegesen az értelmezőt közvetlen végrehajtási módba, a

)) pedig visszatér a definiciós üzemmódba. Ezeknek a szavaknak a felhasználásával a fenti példánk a következőképpen alakul:

```
: 8+ (( 1 3 + 2 * )) LITERAL + ;
```

Ez a definició leforditott állapotban teljesen azonos a

```
: 8+ 8 + ;
```

szóval. A LITERAL és a ((IMMEDIATE szavak. (Miért?)

FREE ("fri") a felhasználható szabad memória byte-jainak számát teszi a stackre.

7. FÜGGELÉK: A FELHASZNÁLÓI SZAVAK JEGYZÉKE

Az alábbi szótár tartalmazza a PRIMO FORTH összes felhasználói szavát, azaz mindazon szavakat, amelyek a FORTH és az EDITOR szótárban vannak. A szavak felsorolása az ASCII rendezési sorrendnek megfelelően történt.

Minden tételnél az első sor a szó által a paraméter stackre kifejtett hatást írja le. A három kötőjel (---) a szó tevékenységét illusztrálja, bal oldalán a stack végrehajtás előtti, jobb oldalán a végrehajtás utáni tartalma áll. (Természetesen csak az érintett szintig ábrázolva.) A stack tetejének mindkét esetben a jobb szélső elem felel meg. A stackben lévő adatok jelölése a következők szerint történt:

JELÖLÉS	ADAT TIPUSA	BITSZAMA	DECIMALIS ÉRTÉKHATARAI
cim	memóriacim	16	0 ... 65535
l	logikai érték	16	0 (hamis) vagy nem 0 (igaz)
igaz	igaz log. érték	16	-1 (eredmény)
hamis	hamis log. érték	16	0
b	tetszőleges byte	8	0 ... 255
kar	megjeleníthető karakter	8	1 ... 151
n	előjeles egész szám	16	-32768 ... 32767
u	előjeltelen egész szám	16	0 ... 65535
d	előjeles dupla pont.	32	-2147483648 ... 2147483647
ud	előjeltelen dupla pont.	32	0 ... 4294967295
sor	screenen belüli sor szám	8	0 ... 31

Amennyiben a kérdéses adat 8 bites, egy fölső 0 byte-tal kiegészitve kerül a stackbe. 32 bites adatnál az előjelet tartalmazó 16 bites érték van felül (a magasabb helyiértékü két byte).

A sor jobb szélén a megfelelő szótár megnevezése áll, ezt P betü előzi meg, ha IMMEDIATE jellegü a szó. A leirás végén a → jel arra utal, hogy egy másik szó definiciójából további információkat nyerhetünk.

! u cim --- FORTH

u eltárolása a cim-en.

ud1 --- ud2 FORTH

Beteszi ud1 utolsó jegyét az output pufferbe (képszerű kiiratás szerkesztése). ud2=ud1/BASE

#> ud --- cim b FORTH

Befejezi a képszerű kiiratás szerkesztését. cim az output puffer címe, b a string hossza.

#BUF --- b FORTH

A memória méretétől függő értéket ad vissza: a rendszer ennyi screen számára biztosit helyet egyszerre a memóriában.

#S ud1 --- ud2 FORTH

Beteszi ud1 jegyeit az output pufferbe, ud2=0 → #

& cim --- u FORTH

A 16 bites u érték felhozatala a cim-ről.

' --- cim FORTH

Használata: ' SZO

cim a kérdéses szó paraméter mezőjének címe (PFA).

(--- P,FORTH

Megjegyzés bevezetése. A megjegyzés végét) jelzi.

((--- P,FORTH

Használata: : SZO ... ((...)) ... ;

Időlegesen visszakapcsol közvetlen végrehajtási módba.

((COMPILE)) --- P,FORTH

Arra utasitja az interpretert, hogy a következő szót akkor is forditsa be, ha az IMMEDIATE típusú. Definicióban használjuk.

)) --- FORTH

Visszakapcsol forditási állapotba. → ((

* n1 n2 --- n3 FORTH

n3=n1*n2. Túlcsordulás esetén hibajelzést ad!

*/ n1 n2 n3 --- n4 FORTH

n4=n1*n2/n3. 32 bites közbenső eredményt használ. → /

*/MOD n1 n2 n3 --- n4 n5 FORTH

n5=n1*n2/n3, maradék n4. → */

+ u1 u2 --- u3 FORTH

u3=u1+u2. Túlcsordulás ellenőrzés nincs.

+! u cim --- FORTH

A cim-en lévő 16 bites érték tartalmához u-t ad. Túlcsordulás ellenőrzés nincs.

+- n1 n2 --- n3 FORTH

n3=-n1, ha n2<0, egyébként n3=n1. Pozitiv n1 esetén tehát n1 felveszi n2 előjelét (=n3).

+LOOP n --- P,FORTH

A ciklusváltozóhoz n-et ad. Amennyiben az nem haladt át a (v-1,v) intervallumon, visszaugrás következik a DO-hoz. Egyébként a végrehajtás a következő szónál folytatódik.

, u --- FORTH

u a szótárstack tetejére kerül. → ALLOT

- u1 u2 --- u3 FORTH

u3=u1-u2. Túlcsordulás ellenőrzés nincs.

--> --- P,FORTH

Az aktuálisnál eggyel nagyobb sorszámú screen betöltése.

-1 --- -1 FORTH

-1 értékű konstans.

-ROLL ... u --- ... FORTH

A felső u+1 elem visszafelé forgatása. 1 -ROLL = SWAP, 2 -ROLL = -ROT. → ROLL

-ROT u1 u2 u3 --- u3 u1 u2 FORTH

A fölső 3 elem visszafelé forgatása.

. u --- FORTH

Az u szám kiírása az aktuális számrendszerben, a lehető legkevesebb jeggyel. A számot egy szóköz követi.

." --- P,FORTH

Használata: ." szöveg"

A szöveg string kiírása.

.R n u --- FORTH

n kiírása u szélességű mezőbe jobbról igazítva.

/ n1 n2 --- n3 FORTH

n3=ent(n1/n2), ahol ent(X) az a legnagyobb egész szám, amely nem nagyobb X-nél. Pl.: 3 2 / 1-et ad, -3 2 / -2-őt. Nullával osztás esetén hibajelzest karunk.

/MOD n1 n2 --- n3 n4 FORTH

n4=ent(n1/n2), n3=n1-n2*n4 az osztás maradéka. → /

0 --- 0 FORTH

0 értékű állandó.

0< n --- l FORTH

l=-1, ha n negativ, egyébként l=0.

0= u --- l FORTH

l=-1, ha u=0, egyébként l=0.

0> n --- l FORTH

l=-1, ha n pozitiv, egyébként l=0.

1 --- 1 FORTH

1 értékű konstans.

1+ u1 --- u2 FORTH

u2=u1+1. → +

1+! cim --- FORTH

A cim-en lévő 16 bites érték tartalmához 1-et ad. → +

1- u1 --- u2 FORTH

u2=u1-1. → -

1-! cim --- FORTH

A cim-en lévő 16 bites érték tartalmát 1-gyel csökkenti.

→ -

2 --- 2 FORTH

2 értékű konstans.

2! ud cim --- FORTH

ud eltárolása a cim-en.

2& cim --- ud FORTH

A 32 bites ud érték felhozatala a cim-ről.

2+ u1 --- u2 FORTH

u2=u1+2. → +

2+! cim --- FORTH

A cim-en lévő 16 bites érték tartalmához 2-őt ad. → +

2- u1 --- u2 FORTH

u2=u1-2. → -

2-! cim --- FORTH

A cim-en lévő 16 bites érték tartalmát 2-vel csökkenti. → -

2-ROT ud1 ud2 ud3 --- ud3 ud1 ud2 FORTH

A fölső 3 duplaszó visszafelé forgatása.

2/ u1 --- u2 FORTH

u2=u1/2. → /

2CONSTANT --- FORTH

32 bites konstans definiálása. → CONSTANT

2DROP ud --- FORTH

32 bites érték eltávolitása a stackről.

2DUP ud --- ud ud FORTH

32 bites érték megkettőzése.

2OVER ud1 ud2 --- ud1 ud2 ud1 FORTH

32 bites érték megkettőzése a stack tetején keresztül.

2ROT ud1 ud2 ud3 --- ud2 ud3 ud1 FORTH

A fölső 3 duplaszó forgatása.

2SWAP ud1 ud2 --- ud2 ud1 FORTH

A stack tetején lévő két 32 bites érték fölcserélése.

2VARIABLE --- FORTH

32 bites változó definiálása. → VARIABLE

3 --- 3 FORTH

3 értékű konstans.

: --- FORTH

Használata: : név ... ;

Új szó definiálása név néven. Ezután a név szó végrehajtása azonos lesz a ... helyén álló szavak végrehajtásának sorozatával. Megjegyzendő még, hogy a : beállítja az aktuális keresési szótárt a definiciós szótárnak megfelelően.

; --- P,FORTH

Definició lezárása, visszatérés közvetlen végrehajtási módba.

;S --- FORTH

Egy screen betöltésének (LOAD) végét jelenti. Általában nincs rá szükség, mert a screen vége automatikusan lezárja a betöltést.

< n1 n2 --- l FORTH

l=-1, ha n1<n2, egyébként l=0.

<# --- FORTH

Képszerű kiiratás szerkesztésének kezdete.

<BUILDS --- FORTH

Használata: : név <BUILDS ... DOES> ... ;

FORTH-típusú objektum definiáló szó fordításidejű viselkedését leíró rész bevezetése.

= n1 n2 --- l FORTH

l=-1, ha n1=n2, egyébként l=0.

> n1 n2 --- l FORTH

l=-1, ha n1>n2, egyébként l=0.

>R u --- FORTH

A paraméter stack tetején lévő szót áthelyezi a return stackre.

? cim --- FORTH

A cim-en lévő 16 bites érték kiiratása. → .

?COMP --- FORTH

Hibaüzenetet ad, ha nincs fordítási állapot.

?DUP u --- u (ha u=0) FORTH

u --- u u (ha u<>0)

16 bites érték feltételes megkettőzése.

?ERROR f b --- FORTH

b sorszámú hibaüzenet adása, ha f igaz.

?EXEC --- FORTH

Hibaüzenetet ad, ha nincs közvetlen végrehajtási állapot.

?KEY --- hamis (nincs bevitel) FORTH

--- kar (van bevitel)

A billentyűzet lekérdezését végzi. Ha talál lenyomott billentyűt, visszaadja a kódját, egyébként hamis logikai

értéket szolgáltat. Nem vár bevitelre!

?LEAVE l --- P,FORTH

Ha l hamis, folytatja a végrehajtást. Ellenkező esetben megfelel a LEAVE szónak. → LEAVE

?STACK --- FORTH

Hibaüzenetet ad, ha stack alul- ill. túlcsordulás van.

?TERMINAL --- l FORTH

l igaz, ha lenyomták a BREAK gombot, egyébként l hamis.

A sor --- EDITOR

Használata: sor A szöveg <RETURN>

A szöveg-et beszúrja az aktuális screen sor-adik sorába, minden ezt követő sor eggyel lejjebb lép, az utolsó pedig elvész. → S

ABORT --- FORTH

Alapállapotba helyezi a futtatórendszert. (Mintha RESET-et nyomtunk volna.)

ABS n1 --- n2 FORTH

n2=abs(n1) abszolútérték képzés.

AGAIN --- P,FORTH

Használata: : név ... BEGIN ... AGAIN ... ;

Végtelen ciklus lezárására szolgál, visszaugrás a BEGIN utáni szó végrehajtására.

ALLOT u --- FORTH

u byte-tal kiterjeszti a szótárstack méretét. Ha nincs a memóriában elegendő hely, hibaüzenetet kapunk.

AND u1 u2 --- u3 FORTH

Bitenkénti logikai ÉS művelet végzése.

AT b1 b2 --- FORTH

A kurzort a b1 sor b2 oszlopába állitja.

AT? --- b1 b2 FORTH

A kurzor a b1 sor b2 oszlopában áll.

B --- EDITOR

A legutoljára megtalált karaktersorozat elejére áll. → F

BASE --- cim FORTH

Annak a rendszerváltozónak a cim-ét adja vissza, amelyik az aktuális számrendszer alapszámát tárolja.

BEGIN --- P,FORTH

Ciklikus vezérlési struktúra bevezető szava, egyben a ciklus kezdetét az őt követő szóban határozza meg. → AGAIN END REPEAT UNTIL WHILE

BL --- szóköz FORTH

A szóköz karakternek megfelelő kódot teszi a stackre.

BLANKS cim u --- FORTH

A cim-től kezdődően u darab szóköz karaktert tesz le.

BOUNDS cim1 u --- cim2 cim1 FORTH

cim2=cim1+u. Stringleirásból DO-ciklus paramétereket állit elő.

BYE --- FORTH

Visszatér a BASIC rendszerhez.

C --- EDITOR

Használata: C szöveg

A szöveg-et bemásolja a szerkesztő kurzor pozíciójába. → M

C! b cim --- FORTH

A cim-re leteszi a b byte-ot.

C& cim --- b FORTH

A cim-ről felhozza a b byte-ot.

C, b --- FORTH

A szótár következő üres helyére beirja a b byte-ot. → ,

C/L --- b FORTH

A képernyőn egy sorba kiférő karakterek száma. (42)

C/LL --- b FORTH

A screen sorainak hossza. (32)

CALL u1 u2 u3 u4 cim --- u5 u6 u7 u8 FORTH

Gépi kódú rutin hivása cim-en. Belépéskor AF=u1, BC=u2, DE=u3, HL=u4, visszatéréskor u5=AF, u6=BC, u7=DE és u8=HL. A szubrutin RET-tel térjen vissza, az alternativ regisztereket, IX és IY regisztereket nem ronthatja.

CFA cim1 --- cim2 FORTH

cim1 egy szó paramétermezőjére, cim2 a kódmezejére mutat.

CLEAR b --- FORTH

A b-edik screen törlése, egyben kijelölése szerkesztésre. Átkapcsol EDITOR módba.

CLS --- FORTH

Le'örli a képernyöt, és a kurzort a bal felsö sarokba állitja.

CMOVE cim1 cim2 u --- FORTH

A cim1-töl kezdve u byte-ot átvisz a cim2-n kezdödö tartományra.

CONSTANT u --- FORTH

Használata: u CONSTANT név

Egy név nevü u értékü konstanst definiál.

COPY b1 b2 --- EDITOR

A b1 sorszámú screen tartalmát átmásolja a b2 sorszámúba.

COUNT cim1 --- cim2 b FORTH

A cim1-en lévö, FORTH szabvány szerint tárolt karaktersorozat cim2 kezdöcimét és b hosszát adja meg.

CR --- FORTH

Új sorba viszi a kurzort.

D sor --- EDITOR

Az aktuális screen sor-adik sorát kitörli a screenböl, de elmenti PAD-re. Minden ezt követö sor eggyel följebb lép, az utolsó pedig szóközökkel lesz tele. → PAD

D+ ud1 ud2 --- ud3 FORTH

ud3=ud1+ud2. Túlcsordulás ellenörzés nincs.

D+- d1 n --- d2 FORTH

d2=-d1, ha n<0, egyébként d2=d1. → +-

D- ud1 ud2 --- ud3 FORTH

ud3=ud1-ud2. Túlcsordulás ellenőrzés nincs.

D->S d --- n FORTH

Dupla pontos szám konverziója egyszeressé. Túlcsordulás esetén hibajelzést ad.

D. d --- FORTH

d kiiratása. → .

D.R d u --- FORTH

d kiiratása u szélességű mezőbe. → .R

D0< d --- l FORTH

l=-1, ha d negativ, egyébként l=0.

D0= d --- l FORTH

l=-1, ha d=0, egyébként l=0.

D0> d --- l FORTH

l=-1, ha d pozitiv, egyébként l=0.

D2/ d1 --- d2 FORTH

d2=d1/2. → 2/

D< d1 d2 --- l FORTH

l=-1, ha d1<d2, egyébként l=0.

D= d1 d2 --- l FORTH

l=-1, ha d1=d2, egyébként l=0.

D> d1 d2 --- l FORTH

l=-1, ha d1>d2, egyébként l=0.

DABS d1 --- d2 FORTH

d2=abs(d1) dupla szavas abszolútérték képzés.

DEC. n --- FORTH

n kiiratása tizes számrendszerben. → .

DECIMAL --- FORTH

Áttérés tizes számrendszerre. → BASE

DEFINITIONS --- FORTH

Az aktuális keresési szótár lesz egyben a definiciós szótár is végrehajtása után.

DELETE b --- EDITOR

b számú karakter törlése a szerkesztő kurzor előtt. → M

DIGIT kar n1 --- n2 igaz (ok) FORTH

kar n1 --- hamis (nem ok)

A kar karaktert megpróbálja n1 számrendszerbeli számjegyként értelmezni. Ha sikerült, visszaadja a megfelelő n2 számot és egy igaz logikai értékkel tér vissza. Ellenkező esetben csak a hamis logikai érték kerül a stackre.

DLITERAL d --- P,FORTH

A fordításidejű d dupla pontos számot futásidőben helyezi a stackre. → LITERAL

DNEGATE d1 --- d2 FORTH

d2=-d1.

DO u1 u2 --- P,FORTH

Ciklusváltozóval vezérelt ciklus nyitó szava, egyben megjelöli az őt követő szót, mint a ciklusmag első utasitását. u1 a ciklusváltozó végértéke, u2 a kezdőértéke. → +LOOP LOOP

DOES> --- cim FORTH

FORTH-típusú objektum definiáló szó futásidejű viselkedését leíró rész bevezetése. A cim a definiált objektum kezdőcíme. → <BUILDS

DPL --- cim FORTH

Egy rendszerváltozó, amelyik számbeolvasásnál megadja a tizedesponttól jobbra eső számjegyek számát. Pont hiányában -1-et ad.

DRAW b1 b2 --- FORTH

Egyenest húz az X=b1 Y=b2 koordinátájú pontig.

DRAWREL b1 b2 --- FORTH

Egyenest húz az X=X+b1 Y=Y+b2 koordinátájú pontig.

DROP u --- FORTH

Eltávolítja a stack tetején lévő értékét.

DUP u --- u u FORTH

Megkettőzi a stack tetején lévő értékét.

E sor --- EDITOR

Az aktuális screen sor-adik sorát szóközökkel tölti fel.

EDISYS --- P,EDITOR

A szerkesztéshez szükséges rendszerszavakat tartalmazó szótár neve. → SYSTEM

EDITOR --- P,FORTH

A szerkesztéshez szükséges felhasználói szavakat tartalmazó szótár neve.

ELSE --- P,FORTH

Feltételes elágaztatás "hamis" ágát vezeti be. → IF

EMIT kar --- FORTH

A kar kódú karakter kiírása a képernyőre.

EMPTY b --- FORTH

A b számú screen által a memóriában elfoglalt helyet felszabaditja. A b screen tartalma elvész!

EMPTY-BUFFERS --- FORTH

A screenek által a memóriában elfoglalt helyeket felszabaditja. → EMPTY

END --- P,FORTH

Az UNTIL régi neve. → UNTIL

ENDIF --- P,FORTH

A feltételes elágaztatás végét jelzi. → IF

ERASE cim u --- FORTH

A cim-től kezdve u byte-ot kinulláz.

ERROR b --- FORTH

b sorszámú hibaüzenet adása.

EXPECT cim b --- FORTH

Maximum b számú karakter beolvasása a billentyüzetről, a szövegbevitelnek megfelelő formában. A karaktersorozatot 0 byte zárja, és a cim-től kezdődően kerül a memóriába.

F --- EDITOR

Használata: F szöveg

A szerkesztő kurzor pozíciójától kezdve keres az aktuális screenben, amig a szöveg-et megtalálja. A kurzort a

megtalált karaktersorozat mögé állítja. Amennyiben a szöveget a screen végéig nem találja, hibaüzenetet ad. → M

FILL cim u b --- FORTH

A cim-től kezdődően letesz a memóriába u darab b byte-ot.

FORGET --- FORTH

Használata: FORGET név

Törli a szótárstackből a név szót és az azt fizikailag követő valamennyi szót. Amennyiben egy nem aktuális szótár kettévágására kerülne sor, hibaüzenetet kapunk.

FORTH --- P,FORTH

Az alap felhasználói szótár neve.

FREE --- u FORTH

u a rendelkezésre álló szabad memóriaterület nagyságát adja byte-okban.

H sor --- EDITOR

Az aktuális screen sor-adik szövegsorát átmásolja a PAD-re, a screen tartalma nem változik. → PAD

H. u --- FORTH

u kiíratása tizenhatos számrendszerben. → U.

HERE --- cim FORTH

A szótárstack következő szabad helyének cim-ét adja vissza.

HEX --- FORTH

Áttérés tizenhatos számrendszerre. → BASE

HOLD kar --- FORTH

Képszerű kiiratás szerkesztésénél a kar kódú karakter behelyezése a kimenő karaktersorozatba. → ⟨# # #⟩

I --- n FORTH

Az aktuálisan legbelső DO-ciklus ciklusváltozóját teszi a stackre.

I sor --- EDITOR

A PAD tartalmát beszúrja az aktuális screen sor-adik sorába. → A

I' --- n FORTH

Az aktuálisan legbelső DO-ciklus ciklusváltozójának végértékét teszi a stackre.

ID. cim --- FORTH

A cim névmezőjű szó nevét irja ki.

IF l --- P,FORTH

Használata: : név ... IF tev.1 ELSE tev.2 ENDIF ... ;

Feltételes elágaztatás bevezető utasitása. Ha l igaz, a tev.1 rész hajtódik végre, egyébként a tev.2, majd mindkét esetben az ENDIF után folytatódik a programvégrehajtás.

IMMEDIATE --- FORTH

A legutoljára definiált szó IMMEDIATE tipusú lesz, azaz definició törzsében is végrehajtódik, és nem kerül lefordításra.

INPUT --- n vagy u FORTH

Beolvas egy egyszeres pontosságú számot a billentyűzetről. Túl nagy szám megadása esetén hibaüzenetet ad!

INPUT. --- ud vagy d FORTH

Beolvas egy dupla pontosságú számot a billentyűzetről.

J --- n FORTH

Az aktuálisan belülről a második mélységű DO-ciklus ciklusváltozóját teszi a stackre.

J' --- n FORTH

Az aktuálisan belülről a második mélységű DO-ciklus ciklusváltozójának végértékét teszi a stackre.

K --- n FORTH

Az aktuálisan belülről a harmadik mélységű DO-ciklus ciklusváltozóját teszi a stackre.

K' --- n FORTH

Az aktuálisan belülről a harmadik mélységű DO-ciklus ciklusváltozójának végértékét teszi a stackre.

KEY --- kar FORTH

A következő begépelt karakter kódját teszi a stackbe. Mindenképpen vár egy beírást!

L --- EDITOR

Az aktuális screent a képernyőre listázza.

LATEST --- cim FORTH

A legutoljára definiált szó névmezejének cim-ét adja vissza.

LEAVE --- P,FORTH

Az aktuálisan legbelső DO-ciklus azonnali elhagyását eredményezi, a végrehajtást a záró +LOOP vagy LOOP után folytatva. → DO

LFA cim1 --- cim2 FORTH

cim1 egy szó paramétermezejének cime, cim2 pedig ugyanennek a szónak a csatolómező cime.

LIST b --- FORTH

A b sorszámú screent a képernyőre listázza, és egyben megjelöli aktuális screenként. Szükség esetén kazettáról tölti be.

LITERAL u --- P,FORTH

Definició belsejében használjuk. A stack fordításidejű legfelső elemét leforditja a szótárstackbe oly módon, mintha közvetlenül begépeltük volna a neki megfelelő értéket.

LL sor --- EDITOR

Az aktuális screen listázása a sor-adik sortól kezdve. → L

LOAD b --- FORTH

A b sorszámú screent betölti, azaz a benne lévő információt beolvassa, mintha a billentyűzetről jönne. → LIST

LOOP --- P,FORTH

*Megfelel az 1 növekményű +LOOP lezárásnak. → +LOOP

M n --- EDITOR

A szerkesztő kurzort n pozicióval (előjelesen) elmozgatja, majd kiírja a kurzort tartalmazó sort, egy | karaktert, valamint a sor számát. A szerkesztő kurzor pozicióját egy _ aláhúzás karakter jelzi. Ez a kiindulópontja a soron belüli szerkesztő utasításoknak.

M* n1 n2 --- d FORTH

d=n1*n2. Kevert pontosságú szorzás.

M/MOD d1 n1 --- n2 n3 FORTH

n3=ent(d1/n1), n2=d1-n1*n3 az osztás maradéka. → /

MAX n1 n2 --- n3 FORTH

n3 az n1 és n2 közül a nagyobbikkal egyenlő.

MIN n1 n2 --- n3 FORTH

n3 az n1 és n2 közül a kisebbikkel egyenlő.

MOD n1 n2 --- n3 FORTH

n3 az n1/n2 osztás maradéka. → /MOD

N --- EDITOR

A legutóbb megtalált karaktersorozat továbbkeresése. → F

NEGATE n1 --- n2 FORTH

n2=-n1.

NFA cim1 --- cim2 FORTH

cim1 egy szó paramétermezejének cime, cim2 pedig ugyanennek a szónak a névmező cime.

NOT u1 --- u2 FORTH

u2 u1 bitenkénti komplemense.

OFF --- FORTH

Fekete "szin" beállitása PLOT-hoz. → ON

ON --- FORTH

Fehér "szin" beállitása PLOT-hoz. → PLOT

ON? b1 b2 --- l FORTH

l igaz, ha az X=b1 Y=b2 koordinátájú pont fehér, egyébként l hamis.

OR u1 u2 --- u3 FORTH

Bitenkénti logikai VAGY művelet végzése.

OUT --- cim FORTH

Rendszerváltozó, amelyet minden EMIT eggyel megnövel. A kiiratást lehet segitségével ellenőrizni. → EMIT

OVER u1 u2 --- u1 u2 u1 FORTH

Stacktartalom megkettőzése a stack tetején keresztül.

P sor --- EDITOR

Használata: sor P szöveg

A szöveg elhelyezése az aktuális screen sor-adik sorában.

A sor korábbi tartalma a PAD-re kerül.

P! b cim --- FORTH

A cim számú portra kiküldi a b byte-ot.

P& cim --- b FORTH

A cim számú portról beolvas egy b byte-ot.

PAD --- cim FORTH

Egy szövegpuffer cim-ét adja vissza. Az EDITOR sorszerkesztő parancsai használják ezt a területet sorok átmeneti tárolására.

PFA cim1 --- cim2 FORTH

cim1 egy szó névmezejének cime, cim2 pedig ugyanennek a szónak a paramétermező cime.

PICK ... u1 --- ... u2 FORTH

Ha a stack legfölső szintjét 0-val jelöljük, mielőtt u1 rákerült volna, akkor az utasitás u1 helyére a stack u1 mélységü elemét ismétli meg. Pl.: 0 PICK = DUP 1 PICK = OVER.

PLOT b1 b2 --- FORTH

A legutoljára kiadott ON vagy OFF utasitásnak megfelelően az X=b1 Y=b2 koordinátájú pontot kigyújtja vagy eloltja. Kigyújtás az ON utasitás hatására következik be, bekapcsoláskor ez az alapállapot.

PROMPT --- FORTH

Használata: PROMPT kar.

Hatására az aktuális keresési szótár bejelentkező karaktere kar. lesz. Amennyiben nem adunk meg karaktert, ez a szótár nem ad bejelentkezést.

R --- u FORTH

A return stack tetejének tartalmát átmásolja a paraméter stackbe.

R sor --- EDITOR

Az aktuális screen sor-adik sorát fölcseréli a PAD-en levő karaktersorozattal. Az eredeti sor a PAD-en megőrződik.

R> --- u FORTH

A return stack tetejének tartalmát átviszi a paraméter stackbe.

REPEAT --- P,FORTH

BEGIN ... WHILE ciklus záró utasítása. Végrehajtásakor a vezérlést a BEGIN utánra adja vissza. → WHILE

ROLL ... u --- ... FORTH

A felső u+1 elem visszafelé forgatása. 1 ROLL = SWAP, 2 ROLL = ROT. u a forgatás során érintett legnagyobb stackmélység. → PICK

ROT u1 u2 u3 --- u2 u3 u1 FORTH

A fölső 3 szó forgatása.

S sor --- EDITOR

A korábbi sor-adik szövegsor a sor+1-edikbe lép, és ugyanígy, ettől lefelé minden sor eggyel lentebb kerül. Az utolsó (31-es) sor elvész, a sor-adik pedig üres lesz.

S->D n --- d FORTH

Egyszeres pontosságú szám dupla pontossá konvertálása.

SAVE b --- FORTH

A b sorszámú screent mágnószalagra menti.

SCR? --- FORTH

Kiírja a képernyőre a memóriában található screenek sorszámát, valamint az első sorukat.

SIGN n d --- d FORTH

Ha n nem negatív, csak a stack tartalom módosul. Egyébként betesz egy mínuszjel karaktert az output

pufferbe. → HOLD

SL --- EDITOR

A szerkesztő kurzort tartalmazó sor környékét listázza ki. → L

SMUDGE --- FORTH

A legutoljára definiált szó "smudge-bitjét" invertálja meg, ezzel megakadályozva vagy lehetővé téve a szó megtalálását kereséskor.

SPACE --- FORTH

Egy szóköz karaktert ir a képernyő aktuális pozíciójába.

SPACES u --- FORTH

u darab szóköz karaktert ir a képernyő aktuális pozíciójától kezdve.

SWAP u1 u2 --- u2 u1 FORTH

A stack felső két elemét megcseréli.

SYSTEM --- P,FORTH

A rendszerszavakat tartalmazó szótár neve. A kezdő programozó számára érdektelen vagy "veszélyes" szavakat találhatjuk itt.

T sor --- EDITOR

A sor-adik szövegsor elejére állitja a szerkesztő kurzort. → M

TASK --- FORTH

Nem csinál semmit, de el lehet felejtetni.

TEST b --- FORTH

A b-edik screent összehasonlitja a memóriában levő azonos sorszámú screennel. Eltérés esetén hibaüzenetet ad.

THEN --- P,FORTH

Az ENDIF másik neve. → ENDIF

TILL --- EDITOR

Használata: TILL szöveg

A szerkesztő kurzortól kezdve a szöveg végéig törli a karaktereket. Ha az adott szöveg nincs a kurzort tartalmazó sorban, hibajelzést kapunk, ekkor a kurzor a következő sor elejére áll.

TOP --- EDITOR

A szerkesztő kurzort a screen első sora első karaktere elé állitja.

TYPE cim b --- FORTH

A cim-től kezdve b darab karaktert ir a képernyőre.

U* u1 u2 --- u3 FORTH

u3=u1*u2. → *

U. u --- FORTH

Előjeltelen szám kiiratása. → .

U/MOD u1 u2 --- u3 u4 FORTH

u4=u1/u2 lefele kerekitve, u3=u1-u2*u4 az osztás maradéka. Nullával osztás esetén hibajelzést ad!

U< u1 u2 --- l FORTH

l igaz, ha u1<u2, egyébként hamis.

U> u1 u2 --- l FORTH

l igaz, ha u1>u2, egyébként hamis.

UD* ud1 u --- ud2 FORTH

ud2=ud1*u. Túlcsordulás esetén hibaüzenetet ad.

UD->S ud --- u FORTH

Konvertálást végez. Hibát jelez, ha a konverzió nem hajtható végre.

UD/MOD ud1 u1 --- ud2 u2 FORTH

Dupla pontos szám osztása 16 bites értékkel, de az eredmény ud2 32 bites. u2 az egyszeres pontosságú maradék.

UD< ud1 ud2 --- l FORTH

l igaz, ha ud1<ud2, egyébként hamis.

UD> ud1 ud2 --- l FORTH

l igaz, ha ud1>ud2, egyébként hamis.

UM* u1 u2 --- ud FORTH

Vegyes pontosságú szorzás. ud=u1*u2

UM/MOD ud u1 --- u2 u3 FORTH

u3=ud/u1 lefele kerekítve, u2=ud-u1*u3 az osztás maradéka. Túlcsordulás esetén hibajelzést ad!

UNTIL l --- P,FORTH

Használata: : név ... BEGIN tev. UNTIL ... ;

Feltétellel vezérelt ciklus. Ha l igaz, az UNTIL után következő szón folytatódik a végrehajtás, egyébként visszaugrás a tev. ciklikus ismétlésére.

VARIABLE u --- FORTH

Használata: u VARIABLE név

u kezdőértékű név nevű változó definiálása. u későbbi végrehajtásakor a tárolt érték címét teszi a stackbe.

VLIST --- FORTH

A szótári elemek nevének kilistázása az aktuális keresési szótártól indulva.

VOC. --- FORTH

Az aktuális definiciós szótár nevét írja ki.

VOCABULARY --- FORTH

Használata: VOCABULARY név

Új szótár definiálása név néven. A név későbbi végrehajtásakor a név szótár lesz az aktuális keresési szótár.

WHILE l --- P,FORTH

Használata: : név .. BEGIN .. WHILE tev. REPEAT .. ;

Feltétellel vezérelt ciklus, elején történő vizsgálattal. Amennyiben l hamis, a vezérlés átadódik a REPEAT mögötti részre, azaz elhagyjuk a ciklust. Egyébként a tev. végrehajtásra kerül, majd a REPEAT-nél visszaugrunk a kezdő BEGIN után.

X --- cim FORTH

Az aktuális grafikus kiiratási pozició X koordinátáját tartalmazó változó címét adja.

X --- EDITOR

Használata: X szöveg

Megkeresi és kitörli a szöveg karaktersorozat következő előfordulását. → M F

XOR u1 u2 --- u3 FORTH

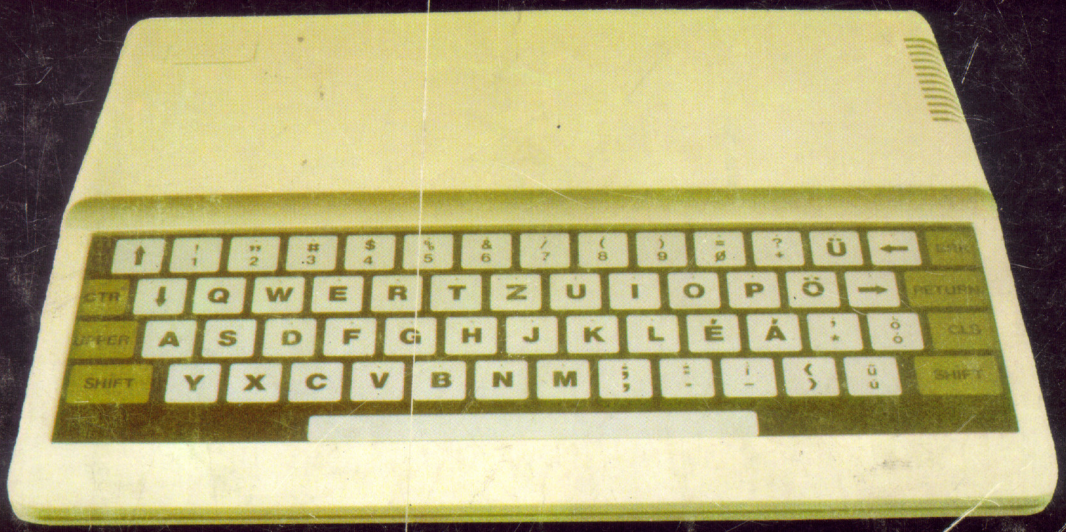
Bitenkénti logikai KIZÁRÓ VAGY művelet végzése.

XYSAVE b1 b2 --- FORTH

X=b1 Y=b2 értékeket állit be (pl.: DRAW-hoz). → X Y

Y --- cim FORTH

Az aktuális grafikus kiiratási pozició Y koordinátáját tartalmazó változó címét adja.



Forgalmazó:
ELEKTROMODUL
1132 Budapest, Victor Hugo u. 13–15.
Telefon: 495-340 ● Telex: 22-5154

Menedzser:
MTA–SZTAKI COSY MŰSZAKI FEJLESZTŐ LEÁNYVÁLLALAT
BUDAPEST 5.
Pf.: 690
1365