BASIC

A **BASIC** (***B*eginner's *A*ll-purpose *S*ymbolic *I*nstruction *C*ode**) általános célú programozási nyelvet 1964-ben készítette Kemény János és Thomas Kurtz a Dartmouth College-ben, oktatási céllal.

Utasítástípusok

Adatkezelés

- LET: értékadás; az érték aritmetikai kifejezés értéke is lehet.
- DATA: listafeltöltés a megadott vagy READ paranccsal beolvasott értékekkel.

Vezérlés

- IF ... THEN ... ELSE: elágazás-képzés.
- FOR ... TO ... {STEP} ... NEXT: ciklusképzés: programblokk adott számú ismétlése a ciklusváltozó értéke szerint
- WHILE ... WEND and REPEAT ... UNTIL: ciklusképzés: programblokk ismétlése, amíg a megadott logikai feltétel igaz értékű; a kiértékelés az első esetben a végrehajtás előtt, a másodikban utána történik.
- DO ... LOOP {WHILE} or {UNTIL}: ciklusképzés: programblokk ismétlése vég nélkül, illetve amíg a megadott logikai feltétel igaz értékű; a kiértékelés az első esetben a végrehajtás előtt, a másodikban utána történik.
- GOTO: ugrás a megadott sorszámú, illetve címkéjű sorra.
- GOSUB: szubrutinhívás: a megadott sorszámú, illetve címkéjű programsorral kezdődő programblokk végrehajtása a RETURN utasításig; ez után a program a hívó utasítást követő sortól folytatódik
- ON ... GOTO/GOSUB: feltételes ugrás, illetve szubrutinhívás, ha a megadott logikai feltétel igaz értékű
- DEF FNX (x) = <aritmetikai kifejezés> függvénydefiníció a program elején; az X függvénynév 1 karakter lehet, az aritmetikai kifejezésben az x változóval végezhet műveleteket; négyzetre emelési minta: `DEF FND(x) = x*x`.

Be-/kimenet

- LIST: a beolvasott kód megjelenítése képernyőn.
- PRINT: üzenet kiírása a képernyőre vagy egy nyomtatóra.
- INPUT: változó értékének beolvasása megadott üzenet kiírása után.
- TAB or AT: a következő karakter pozícionálása a képernyőn vagy nyomtatón.

Egyéb műveletek

- REM: megjegyzés; legtöbbször egy programrész azonosítására, vagy címkeként használatos.
- USR: vezérlés-átadás gépi kódú szubrutinra, ami rendszerint egy alfanumerikus karakterlánc, DATA utasítások sorozata.
- TRON: nyomkövetés a képernyőn a végrehajtott utasítások sorszámának kiírásával, belövési vagy hibakereséséi céllal (Lásd TRON command).
- TROFF: nyomkövetés vége a TRON utasítás után..

Adattípusok és változók

A BASIC nyelvjárások nagy részében nem kell a változókat deklarálni. Az új változókat ebben az esetben a parancsértelmező futásidőben hozza létre a nevük első előfordulásakor. Ez hibalehetőséget jelent annyiban, hogy az elgépelt nevű változók is létrejönnek így. Az idők folyamán azonban több megoldás is létrejött:

Az első változatban, a Dartmouth Basic-ben egy, illetve kétjegyű változóneveket használhattunk, az első jegy betű (A-Z), az (opcionális) második számjegy (0-9), így a BASIC összesen 11x26=286 változót tudott kezelni (A-Z, A0-A9,…, Z0-Z9).
Az értékadás a LET kulcsszóval történt (például LET x=26), ami több későbbi változatban is benne maradt, bár idővel fölöslegessé válva kikopott.
A változók A Dartmouth Basicben mind lebegőpontos számok voltak, később megjelentek más típusok is. Ezeket általában a változónév végén álló karakterek jelzik:

- % - egész szám (például a%=5)
- ! vagy nincs – (egyszeres pontosságú) lebegőpontos szám (például a=5.1)
- # – kétszeres pontosságú lebegőpontos szám (például a#=5.5)
- $ – szöveges változó (string) (például a$="Cica")

Megjegyzendő, hogy a stringeket kezelő függvények neve is $-re végződik általában.
Egyes verziók lehetővé tették a változók típusának előzetes deklarációját, a DEFINT, DEFSTR, … utasításokkal:

```
DEFINT A-Z: REM minden változó legyen egész
DEFSTR S: REM az S-sel kezdődő változók stringek
```

Idővel ez a jelölésmód korszerűtlenné vált, rontotta a nyelv használhatóságát. Újabb értelmezőkben - mint a Quick Basic vagy a Visual Basic - elterjedt az a megoldás, hogy a tömbök deklarására használt DIM parancsot lehet használni. Példa (Visual Basic):

```
Dim x As Long
Dim y As String
x=123456
y="Cica"
```

A "klasszikus" Visual Basic bár nem tette kötelezővé a változók deklarálását, az Option Explicit fordítási direktíva bekapcsolásával kötelezhette magát erre a felhasználó. A Visual Basic .NET a CLI-kompaibilitás miatt nem támogatja a deklarálatlan változókat.
Ez eredeti változatban az egyetlen használható típus-konstrukció a tömb, a tömböket a DIM utasítással lehet deklarálni ('dimenzionálni'), egyes változatok a dimenzionálatlan tömböket automatikusan tízelemű tömbként dimenzionálták, pl:

```
DIM x(22): REM szabályos dimenzionálás
LET y(3)=3: REM implicit dimenzionálás: DIM y(10), nem minden verzióban
```

A tömbindexek általában eggyel kezdődtek, egyes megvalósításokban viszont nullától (ekkor a tömb a megadottnál eggyel több elemet tartalmazott), vagy az 'OPTION BASE {0|1}' utasítással volt beállítható a kezdőindex.

Sorszámozás

A korai BASIC változatokban kötelező volt a sorok számozása. Minden programsor elején egy sorszám állt. Ennek több oka volt:
1. Az ugrásoknál (GOTO, GOSUB) sorszámra hivatkozva lehetett csak megadni, hova ugorjon a vezérlés, a nyelv kezdeti változatai nem ismerték a címkéket.
2. A kezdetleges fejlesztőkörnyezeteknél a sorszámról döntötte el az értelmező, melyik programsort írta be a felhasználó. Ha például a 30-as sor után írta be a 20-ast, akkor az sorrendben a 30-as elé került futtatáskor.
3. A hibaüzeneteket is a sorszámot kiírva adták meg az interpreterek.

A sorszámozás hátránya a nehézkes bővíthetőség volt: ha a sorokat 10-esével számoztuk, akkor a 20-as és 30-as sor közé csak 9 sort szúrhattunk be. Emiatt a nyelvből lassan eltűntek a sorszámok, bár sokáig lehetett használni őket a visszafelé kompatibilitás megőrzése miatt. Microsoft vonalon az 1998-ban megjelent Visual Basic 6 volt az utolsó nyelv, ami még támogatta a használatukat. A feltétel nélküli ugrásokhoz a nyelv címkéket használ, amik sor elején álló, kettőspontra végződő azonosítónevek. A címkéket nem kell deklarálni.

Példa sorszámos ugrásra:

```
10 PRINT "Ez a szöveg ismétlődni fog!"
20 GOTO 10
```

Példa a címke használatára

```
eleje:
PRINT "Ez a szöveg ismétlődni fog!"
GOTO eleje
```

Példa üzenet kiírására

Az interpretált BASIC parancsként tudja futtatni a "PRINT" *(írd ki)* utasítást:

```
PRINT "Hello World!"
```

Megj.: A PRINT sok nyelvjárásban rövidíthető kérdőjellel:

```
? "Hello World!"
```

Programba illesztve (klasszikus változat, sorszámok használatával):

```
10 PRINT "Hello World!"
20 END
```

Megjegyzés: Az "END" *(vége)* utasítás opcionális, a programfutás az utolsó sornál mindenképp véget ér.

Futtatása:

```
RUN
```

Példaprogramok

Példa 1: ***Strukturálatlan***, eredeti BASIC
(Applesoft BASIC nyelv)

```
  10 INPUT "Kérem adja meg a
nevét: "; U$
  20 PRINT "Üdvözlöm "; U$
  25 REM
  30  INPUT  "Hány  csillagot
szeretne? "; N
  35 S$ = ""
  40 FOR I = 1 TO N
  50 S$ = S$ + "*"
  55 NEXT I
  60 PRINT S$
  65 REM
  70    INPUT    "Kér    még
csillagot? "; A$
  80 IF LEN(A$) = 0 GOTO 70
  90 A$ = LEFT$(A$, 1)
 100  IF  (A$  =  "i")  OR  (A$  =
"I") THEN GOTO 30
 110   PRINT   "Viszontlátásra!
";
 120 FOR I = 1 TO 200
 130 PRINT U$; " ";
 140 NEXT I
 150 PRINT
```

Példa 2: Modern ***Strukturált*** nyelv

```
  INPUT  "Kérem  adja  meg  a
nevét: "; UserName$
  PRINT     "Üdvözlöm     ";
UserName$
  DO
     INPUT   "Hány   csillagot
szeretne?"; NumStars
     Stars$ = ""
     Stars$   =   REPEAT$("*",
NumStars)   ' <- ANSI BASIC
     ''--or--''
     Stars$                =
STRING$(NumStars, "*")    ' <-
MS   BASIC
     PRINT Stars$
     DO
        INPUT    "Kér    még
csillagot? ";  Answer$
     LOOP UNTIL Answer$ <> ""
     Answer$  =  LEFT$(Answer$,
1);
  LOOP  WHILE   UCASE$(Answer$)
= "Y"
  PRINT "Viszontlátásra! ";
  FOR I = 1 TO 200
     PRINT UserName$; " ";
  NEXT I
  PRINT
```