NAGYPONTOSSÁGÚ EGÉSZ-ARITMETIKA
EXCEL VISUAL BASIC KÖRNYEZETBEN

SZLÁVI PÉTER

2013-2016

TARTALOM

0. A FELADAT ÉS A CÉL

Feladat

… nagypontossággal számolni az (pozitív) egész számok körében. A megvalósítandó műveletek:

- Eggyel növelés és csökkentés
- Összeadás
- Kivonás
- Szorzás
- Maradékos osztás: egész hányados és maradék
- Azonosság reláció
- Rendezési reláció
- Konverziók (String↔nagypontosságú egész)[1]

Cél

A feladatkör alkalmas arra a **hétköznapi algoritmusaik** miatt, hogy bevessük az **algoritmizálás és a kódolás gyakorlásá**ra. Mivel amúgy is nagy óraszámban kerül szóba az Excel **táblázatkezelő**, ezért könnyen becsempészhető az anyagba e feladatkör, azaz némi programozás. Ez a cél alapvetően befolyásolja majd az adatok **ábrázolás**át.

A haladás vázlata

Tisztázandók az alábbiak:

- A nagypontosságú egészek ábrázolása.
- A Visual BASIC (VB) nyelvi legfontosabb elemei.
- A VB for Application (VBA) Excelhez illeszkedő, szükséges elemei, Excel programozási tudnivalók. (Vannak apróbb eltérések a VB-től még a nyelv Excelhez nem kötődő elemeit tekintve is.)
- Néhány feladatfüggő kódolási konvenció. Speciális kódolási trükk.
- A kívánt műveletek nagyvonalú algoritmusa, a „manuális módszerre” építve.

---

[1] Szokásosan segítségünkre lesz az adott típus, most a nagy-egészek típusába sorolt adatok I/O-jánál.

1. AZ EGÉSZ SZÁMOK ÁBRÁZOLÁSA

A nagypontosságú egészek számos módon ábrázolhatók. Lásd az előadás anyagát! Mivel az egész számok **előjel**ből, esetlegesen **számrendszer** megadásából és a **számjegyei**ből, ezért az alábbi ábrázolást választjuk (majdan Excel környezetben kódolva):

```
Típus
  TElőJel={"-","","+"}     [→TElőjel⊆Szöveg]
  TSzámjegyek=Szöveg       [értsd: számjegyek sorozata⊆Szöveg]
  TSzámrendszer=Szöveg     [l. alább]
```

Ezekből tákoljuk össze a nagy-egészeket. Tehát minden nagy-egészet egy **szöveg**gel ábrázolunk:

```
  TNE=Szöveg   [TNE = Nagy-Egészek típusa]
```

Az alábbi séma teljesülését várjuk el (és garantáljuk):

`TNE=TElőjel & TSzámjegyek & TSzámrendszer`[2]. (TNE1)

Mivel „szokás” a pozitív előjelet nem kiírni, ezért ilyen esetben megengedjük az előjel elhagyását. Ezt megengedi a `TElőjel` definíciója is. Szintén a szokásokra hivatkozva a 10-es számrendszer jelölése is elhagyható, azaz az alábbi séma is megengedett:

`TNE=TElőjel & TSzámjegyek`. (TNE2)

A `TSzámrendszer`, ha jelölve van, akkor felismerhetőnek kell lennie! Ezért rá az alábbi séma kell teljesüljön: `"(" & TDecimálisSzámjegyek & ")"`. A `TSzámjegyek` típusa a szokásos karakterek sorozatát tartalmazó típus, azaz az elemek a `"0"`, `"1"`, …, `"9"` (∈`TDecimálisSzámjegyek`), `"A"`, `"B"`, …, `"F"`, sőt folytatható egészen `"Z"`-ig. Ezzel a nem hétköznapi általánosítással akár a 36-os számrendszert is használhatónak kínáljuk föl, ui. a 36-os számrendszerben a legnagyobb számjegy éppen a `"Z"`.

Például legális `TNE`-beli értékek: `"0"`, `"-1"`, `"+1(2)"`, `"-101(2)"`, `"FF(16)"`, `"-1234A(12)"`.

---

[2] A szövegek közötti konkatenáció (=egymásutánírás) műveletét –a VB hagyományt alkalmazva– a „&” jellel jelöljük.

2. A VB(A) LEGFONTOSABB NYELVI ELEMEI

Ez a rész csak emlékeztető jellegű. Azoknak szól, akik még nem programoztak VB-ben. Alapvető programozási ismeretek birtokában elegendő néhány nyelvi elem szintaktikai összefoglalása.

Értékadás

Algoritmikusan	VBA-ban
vált$_1$:=kif$_1$; vált$_2$:=kif$_2$; … [A sor végén nincs elválasztó jel.]	vált$_1$ = kif$_1$: vált$_2$ = kif$_2$: … ’ A sor végén nincs elválasztó jel.

A megjegyzés szintaxisa is kiderül a fenti átírási szabályból!

Elágazások

Algoritmikusan	VBA-ban
Ha feltétel **akkor** utasítások	**If** feltétel **Then** utasítások

Ha feltétel **akkor**
  utasítások
különben
  utasítások
Elágazás vége
[a **különben**-ág elhagyható]
	If feltétel **Then**
	  utasítások
	Else
	  utasítások
	End If
	’ az **Else**-ág elhagyható

Elágazás
  feltétel$_1$ **esetén**
    utasítások$_1$
  feltétel$_2$ **esetén**
    utasítások$_2$
  …
  egyéb esetben
    utasítások
Elágazás vége
[az **egyéb**…-ág elhagyható]
	If feltétel$_1$ **Then**
	  utasítások$_1$
	ElseIf feltétel$_2$ **Then**
	  utasítások$_2$
	…
	Else
	  utasítások
	End If
	’ az **Else**-ág elhagyható

Ciklusok

Algoritmikusan	VBA-ban

Ciklus i=K-**tól** N-**ig**
  utasítások
Ciklus vége
	For i = K **To** N
	  utasítások
	Next i

Ciklus i=K-**tól** N-**ig** L-**esével**
  utasítások
Ciklus vége
	For i = K **To** N **Step** L
	  utasítások
	Next i

```
Ciklus amíg feltétel
  utasítások
Ciklus vége
```

```
Do While felétel
  utasításaok
Loop

        vagy

While feltétel
  utasítások
Wend
```

```
Ciklus
  utasítások
amíg feltétel
Ciklus vége
```

```
Do
  utasítások
Loop While feltétel
```

Alprogram-definiálás

Algoritmikusan

```
Eljárás Elj(paraméterek):
  utasítások
Eljárás vége.

[Ha nincs parméter, a zárójelek
 elhagyhatók! A paraméterek „,”-vel
 választandók el!]
```

VBA-ban

```
Sub Elj(paraméterek)
  utasítások
End Sub

' Ha nincs parméter, a zárójelek
' akkor is kellenek! A paraméterek
' „,”-vel választandók el!
```

Algoritmikusan

```
Függvény Fv(paraméterek):Típus
  utasítások
  Fv:=kif
Függvény vége.

[L. előbbi megjegyzést!]
```

VBA-ban

```
Function Fv(paraméterek) As Típus
  utasítások
  Fv = kif
End Function

' L. előbbi megjegyzést!
```

Példa:

```
Eljárás SzétszedS(Konstans egesz:Szöveg, Változó alap:Egész, ej:Szöveg,
                  hossz:Egész, egesz2:Szöveg):
  …
Eljárás vége.
```

VB megfelelője:

```
Sub SzétszedS(egesz As String, alap As Integer, ej As String, _
              hossz As Integer, egesz2 As String)
  …
End Sub
```

Megjegyzések a fenti példához:

1) Egy utasítás eltörhet több sorra. Hogy van még az utasításnak folytatása, azt a sorvégére tett „_”-jellel kell jelezni.

2) A paraméteráradás ún. **címszerinti**, aminek a révén a paraméterek akár módosulhatnak is a szubrutin végrehajtása során. Olyankor, amikor ki akarjuk fejezni azt, hogy az adott paraméter csak „befelé szállíthatja” az értéket a szubrutinba, akkor elé ki kell írni a `ByVal` kulcs-szót, amely szó

szerint megfelel a szakirodalombeli **érték szerinti** paraméterátadásnak. Sejtvén a fenti eljárás célját, így is írhajuk a fejsorát:

```
Sub SzétszedS(ByVal egesz As String, alap As Integer, ej As String, _
                   hossz As Integer, egesz2 As String)
```

Alprogram-hívás

Algoritmikusan	VBA-ban

`Elj(akt.paraméterek)`
`[Ha nincs parméter, a zárójelek elhagyhatók! A paraméterek „,”-vel választandók el!]`
	`Call Elj(akt.paraméterek)`
	`’ Ha nincs parméter, a zárójelek`
	`’ elhagyhatók! A paraméterek „,”-`
	`’ vel választandók el!`

`…Fv(akt.paraméterek)…`	`…Fv(akt.paraméterek)…`

Példák:

```
Call SzétszedS(e1, alap, ej, h, e2)
```

vagy

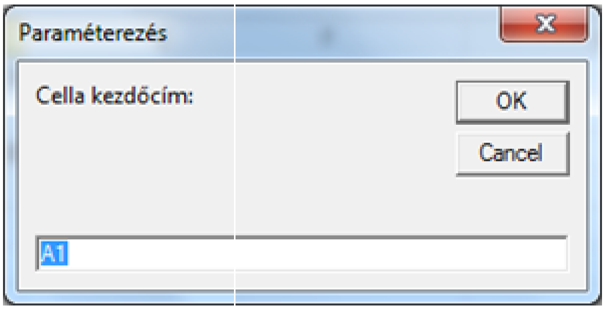
```
Call SzétszedS("-1010(2)", alap, ej, h, e2)
```

Input/output

Minimális szükség lesz a műveletek e fajtáira. Megelégszünk az alábbi két lehetőséggel. Mindkettő szokásos GUI-művelet, amely egy külön ablak megnyitásával végzi el a kellő kommunikációt. Elsőként a bevitel módját nézzük meg!

<table>
<tr><th>Algoritmikusan</th><th>VBA-ban</th></tr>
<tr><td><code>Be: változó</code></td><td><pre>változó = InputBox(Kérdés, _
                   AblakCím, _
                   DefaultÉrték)
’ a 2., 3. paraméter elhagyható! [3]</pre></td></tr>
</table>

Példa:

<table>
<tr><th>VB-kód</th><th>Eredménye a képernyőn</th></tr>
<tr><td><pre>hovaS = InputBox(_
     "Cella kezdőcím:", _
     "Paraméterezés", _
     "A1")</pre></td><td>

Cella kezdőcím:
A1</td></tr>
</table>

---

[3] L. https://msdn.microsoft.com/en-us/library/office/ff839468.aspx

Másodjára lássuk a kiírás módját!

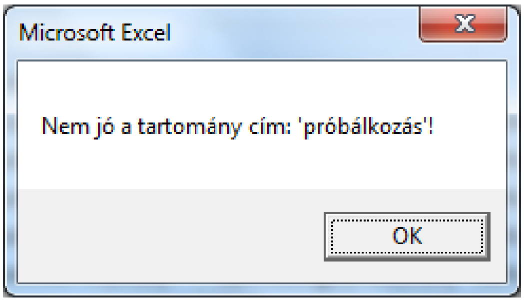
Algoritmikusan	VBA-ban
Ki: kif [a kif tetszőleges típusú formula]	**MsgBox**(kifS) ' a kifS a kif stringes párja; ' az átalakítás általában konkatenációkat és konverziókat kíván!

Példa:

VBA kód

```
' tegyük föl, hogy a hovaS-ben a
' "próbálkozás" szöveg található
MsgBox ( _
    "Nem jó a tartomány cím: '" & _
    hovaS & "'!")
```

Eredménye a képernyőn



Nem jó a tartomány cím: 'próbálkozás'!

Néhány fontos további kulcs-szó

VBA kulcs-szó	Magyarázat
Típusok	
Integer **String** **Boolean** **Variant**	Egész Szöveg Logikai Tetszőleges (ún. dinamikus) típus
Rekord-típus	
Type T mező$_1$ **As** T$_1$ mező$_2$ **As** T$_2$ … **End Type**	Létrejön egy új rekord-típus T néven, a leírásban szereplő mezőnevekkel és hozzájuk tartozó típusokkal.
…	
Dim r **As** T	Helyet foglalunk az r T típusú rekord-változónak; azaz deklaráljuk.
r.mező$_1$ = …	Az r rekord mező$_1$ mezője értéket kap.

Deklarációk, tömb-deklarációk

Dim változó **As** T	T típusnyi helyet foglal a változónak (tehát nem csak tömbnek!).
Dim Tömb(db) **As** T	Helyet foglal db darab T típusú elemnek, Tömb néven, amelyet 0-tól indexeli.
Dim Tömb(tól **To** ig) **As** T	Helyet foglal T típusú elemeknek, annyinak, hogy az indexek tól..ig fussanak.

Szöveg függvények

szöveg$_1$ **&** szöveg$_2$	Egymáshoz fűzi a szövegeket
Len(szöveg)	Hossz
Left(szöveg, db)	Eleje (db≥0)
Mid(szöveg, tól, db)	Rész-szöveg (tól>0; db≥0)
Right(szöveg, db)	Vége (db≥0)
InStr(mibenSzöveg, miSzöveg)	Szövegben elölről keres
InStrRev(mibenSzöveg, miSzöveg)	Szövegben hátulról keres
UCase(szöveg)	Nagybetűsít
LCase(szöveg)	Kisbetűsít
Val(szöveg)	Számmá konvertál
Str(szám)	Szöveggé konvertál (elején szóköz)
Trim(szöveg)	Nyitó és záró szóközöket elhagyja
LTrim(szöveg)	Nyitó szóközöket elhagyja
RTrim(szöveg)	Záró szóközöket elhagyja
Chr(szám)	A szám kódú karakter
Asc(jel)	A jel karakter kódja

Aritmetikai műveletek

+, -	Additív műveletek
***, /**	Valós értékű multiplikatív műveletek
**Mod, **	Egész értékű multiplikatív műveletek
^	Hatványozás

Logikai műveletek

And	És
Or	Vagy
Not	Nem

A továbbiakhoz ajánlom az alábbi információforrásokat:

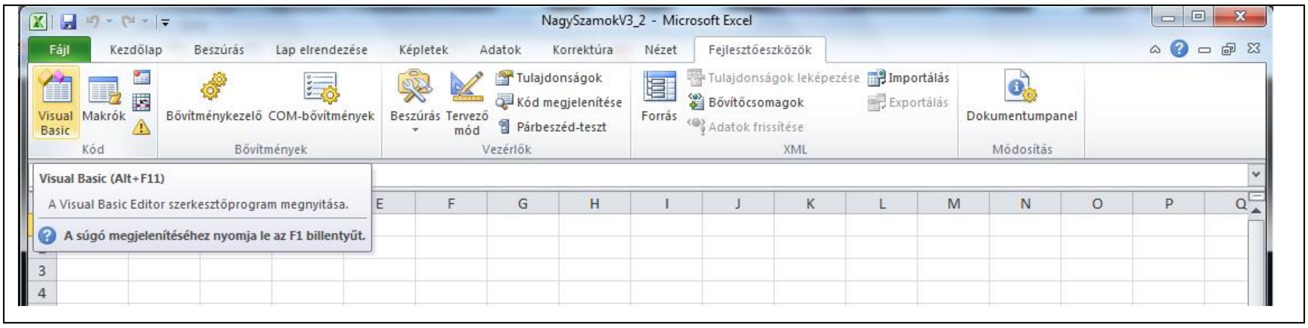
- az Excel súgója
- az MSDN VB referenciája (http://msdn.microsoft.com/en-us/library/sh9ywfdk%28v=vs.80%29.aspx)

3. EXCEL VBA EGYSZER-EGY

Itt is csak a legfontosabb információk átadására szorítkozunk.

Kezdet – alapinformációk

Az Excel VBA-jában történő programíráshoz először meg kell nyitni a **kódmodul**t (Modul1-et):



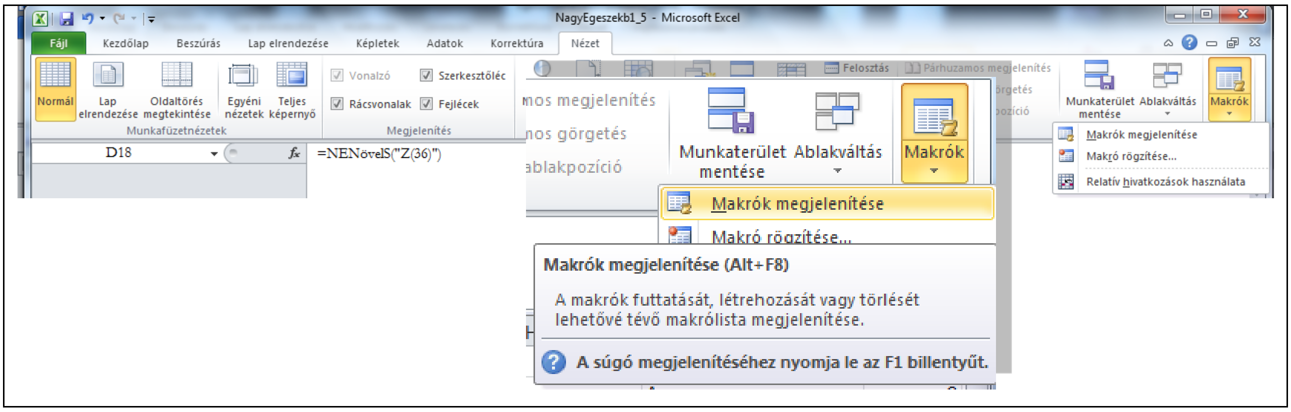
Visual Basic (Alt+F11)

A Visual Basic Editor szerkesztőprogram megnyitása.

A súgó megjelenítéséhez nyomja le az F1 billentyűt.

A VBA fejlesztőkörnyezet megnyitása után illesszünk be egy kódmodult (**`Insert`** / **`Module`** menü)! Így létrejön a Module1 kódmodul, ahova kell majd írnunk a saját függvényeinket és eljárásainkat.

Ha a **`Fejlesztőeszközök`** menücsoport nem látszik a szalagon[4], akkor a „makrózásra” kell rávenni az Excelt: **`Nézet`** / **`Makrók`** / **`Makrók megjelenítése`**, majd egy makrónév beírása után **`Létrehozás`** menü. A makrónév ilyenkor tetszőleges, mert csak azért írtuk be, hogy létrehozzuk a kódmodult. (Magát a létrehozott makrót, amint bejutottunk a kódmodulba, törölhetjük.) L. alább:



D18 =NENövelS("Z(36)")

Makrók megjelenítése (Alt+F8)

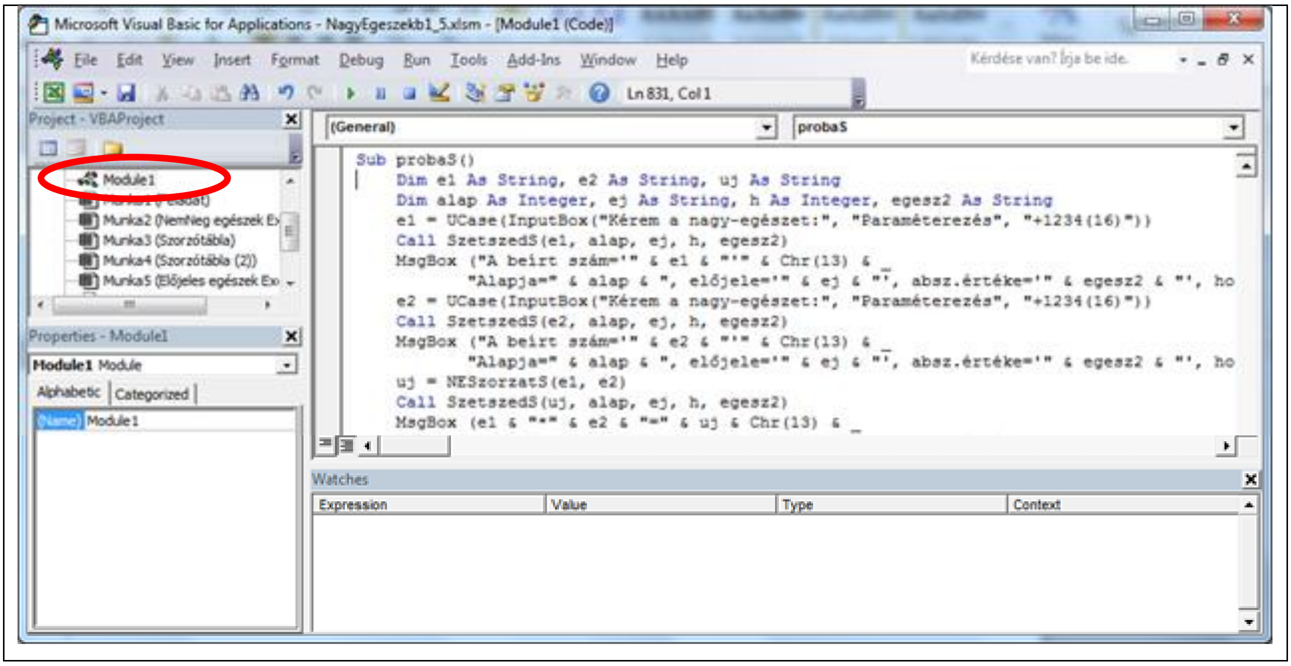
A makrók futtatását, létrehozását vagy törlését lehetővé tévő makrólista megjelenítése.

A súgó megjelenítéséhez nyomja le az F1 billentyűt.

A kódszerkesztő ablak az alábbi szerkezetű. Mint alább látható: a futtatás (**`Run`** menü) mellett eszközök vannak a nyomkövetésre (**`Debug`** menü). Ha már számos alprogramot beírtunk, akkor egy, az ablakban nem látható kiválasztását segíti a szerkesztő ablak tetején jobbra látható lenyíló menü.

---

[4] A hiányzó menücsoport a szalagra feltehető: (`Fájl` csoportban) a `Beállításokon` belül található a `Menüszalag testreszabása` rész, itt kell a `Fejesztői eszközöket` „bepipálni”.



```
Sub probaS()
    Dim e1 As String, e2 As String, uj As String
    Dim alap As Integer, ej As String, h As Integer, egesz2 As String
    e1 = UCase(InputBox("Kérem a nagy-egészet:", "Paraméterezés", "+1234(16)"))
    Call SzetszedS(e1, alap, ej, h, egesz2)
    MsgBox ("A beirt szám='" & e1 & "'" & Chr(13) & _
            "Alapja=" & alap & ", előjele='" & ej & "', absz.értéke='" & egesz2 & "', ho
    e2 = UCase(InputBox("Kérem a nagy-egészet:", "Paraméterezés", "+1234(16)"))
    Call SzetszedS(e2, alap, ej, h, egesz2)
    MsgBox ("A beirt szám='" & e2 & "'" & Chr(13) & _
            "Alapja=" & alap & ", előjele='" & ej & "', absz.értéke='" & egesz2 & "', ho
    uj = NESzorzatS(e1, e2)
    Call SzetszedS(uj, alap, ej, h, egesz2)
    MsgBox (e1 & "*" & e2 & "=" & uj & Chr(13) & _
```

Az itt definiált **függvények** az Excel számolótábláinak celláiba beírhatók, a standard függvényekhez hasonlóan. A paraméterei a számolótáblában megszokott **pontosvessző**vel választandók el, dacára annak, hogy a VBA programban **vessző** az elválasztó jel.

Fontos tudni, hogy **a függvények (`Function`) a tábla celláit nem módosíthatják**, csak annak az egy cellának a tartalmára lehetnek hatással, amelyhez tartozó képletében szerepelnek! **Eljárások (`Sub`)** a táblát korlátozás nélkül **módosíthatják**.

Cellák

Tartománycím típusa: **`Range`**. (A cella az egyetlen cellát tartalmazó tartomány!) A tartomány egy objektum-osztály[5], így vannak jellemzői (=property=attribútum) és rávonatkozó műveletei (=metódus). Néhány fontos jellemzője, művelete:

- a „szokásos” (`$A$1` alakú) címe **`.Address`**, amely lekérdezhető;
- a cellái: **`.Cells`**, amely lekérdezhető;
- a celláinak száma: **`.Cells.Count`**, amely lekérdezhető;
- kijelölése: **`.Select`**, amely végrehajtható;
- aktívvá tétele: **`.Activate`**, amely végrehajtható…

Az alábbi példában az `A10:D20` tartomány bal-felső 4×2-es (4 soros, 2 oszlopos) részét kitölti:

```
Dim s As Integer, o As Integer ' s: sor, o: oszlop
For s = 1 To 4
  For o = 1 To 2
    Range("A10:D20").Cells(s, o) = s * o ' s,o relatív koordináták!
  Next o
Next s
```

---

[5] Az osztály szót OOP-s (objektum-orientált programozás) értelemben használtuk. E megjegyzéssel elsősorban a szintaxisára utalunk: `objektum.jellemző`, `objektum.művelet`.

Feladat:

Nyissa meg a VBA kódablakát, majd az előbbi kóddarabot egy **próba** szubrutinba foglalva gépelje be. (**Nézet** / **Makrók** / **Makrók megjelenítése**, Makrónév: „próba” / **Létrehozás**.) Nyomkövesse a **Debug** menübeli **Step into** (**F8**) funkció használatával, s közben figyelje a táblázat lapjának változását is! Azt is vegye észre, hogy külön kérés nélkül is hajlandó megmutatni a fejlesztőkörnyezet az **s** és **o** változók pillanatnyi tartalmát! Mennyi az **o** értéke a **Next** o után?

A VBA programban létrehozható **tartománycím** típusú változó (amely szintén objektum!):

```
Dim hova As Range ' helyet foglalunk egy tartománycím típusú változónak
Set hova=Range("D5:E6") ' hova-nak beállítjuk a címtartományát
hova.Select ' a táblában kijelöljük a "D5:E6" tartományt
hova.Cells(1, 1).Value = "szöveg" ' a D5 tartalma a "szöveg" lesz
```

Megjegyzés: egy VBA-beli **objektum**ra (tehát egy nem „mezei” VB típusú változóra) vonatkozó értékadás a **Set** művelettel végezhető csak el!

Feladat:

Illessze be ezt a kóddarabot is az előbbi szubrutin végére, és nyomkövetve próbálja ismét ki! Tegyen töréspontot a **Set** utasításhoz (pl. a sortól balra az ablakba kattintva, vagy a sorra állva **F9**)! Indítsa el (a ▶ lejátszás gombot megnyomva; vagy **F5**), figyelje az újdonsült kóddarab műveleteinek hatását a számolótáblában!

Feladat:

Definiáljon egy „cellamásoló” függvényt! Fejsora lehet az alábbi:

`Function Másol(honnan As Range) As String`.

Próbálja ki úgy, hogy ennek hívását –számos helyen– beilleszti az Excel számolótáblájába!

Mi történik a `D10=Másol(E10:F12)` nyilvánvalóan értelmetlen használat során?

Feladat:

Kísérletezgetve derítse ki, a **Range.Cells** objektumnak milyen tulajdonságát kérdezhetjük le a **.Column**, ill. a **.Row** az ún. property-jével! [6]

Speciális kódolási trükkök

Ha egy cellatartomány **minden** cellájára szeretnénk ciklusban megvalósítható módon hivatkozni, akkor az alábbi a legegyszerűbb megoldás.

```
Sub MitCsinál()
    Dim cim As String: cim = "B11:D15"
    Dim tart As Range: Set tart = Range(cim)
    Dim c As Range
    For Each c In tart
      c.Select: c = c.Row & "," & c.Column
    Next c
End Sub
```

---

[6] A bal-felső cellájának oszlop-, ill. sor-koordinátáját (sorszámként).

Ugye nagyszerű ez az új ciklus-szervező utasítás! Hogy valósíthatná ugyanezt meg a „hagyományos" `For`-utasítással?

Feladat:

Próbálja ki (nyomkövetéssel és a `c` változó tartalmának figyelésével), majd magyarázza el a működést!

Hogyan változik az eredmény, ha a

```
c = c.Row & "," & c.Column
```

utasítást kicseréli erre:

`c = c.Address` ?

Függvényeink, eljárásaink **biztonság**os működése megkívánja, hogy a baj elkövetésekor kézben tartsuk a programunkat, illetve ha a baj elkerülhetetlen, akkor is magunknál tartsuk a vezérlést. Ilyen hibák fordulhatnak elő:

- egy kért tartomány címe hibás (pl. formailag),
- egy paraméter nem olyan értékű vagy alakú, mint várnánk,
- …

Hibák figyelése és lekezelése:

- bizonyos hibák (pl. tartománycím szintaktikai hiba) elkerülhetetlenül futási hibát okoz; ilyenkor láthatjuk a cellában az „`#ÉRTÉK!`" üzenetet; hogy ezt mégis elkerüljük, a VBA-ban van egy „ősi" utasítás, az „`On Error Goto` *`címke`*"; ha ezt az utasítást az eljárás elején végrehajtatjuk, akkor ezután bármely hiba esetén a *`címké`*vel jelzett részre kerül a vezérlés, hogy ott magunk döntsünk a teendőkről;
- vannak hibák, amelyek látványos hibát nem okoznak, „csak" rossz eredményre vezetnek; ilyenkor az egységes kezelés kedvéért, amikor a hibát felfedeztük, mi magunk „generálunk" hibát, ami által –hasonlóan az előzőhöz– a hibakezelő részbe irányítjuk a vezérlést.

Példa:

```
Sub Eljárás()
    Dim hovaS As String, hova As Range, db As Integer
    On Error GoTo Hibak ' minden hiba a „Hibak" cimkéjű részhez vezet
    ' itt van valami - 1
    Set hova = Range(hovaS) ' itt keletkezhet futási hiba (438 kóddal),
                            ' ha a hovaS-ben fals cím található
    ' itt van valami - 2
    If db = 0 Then ' elfogadhatatlan db érték
      Err.Raise (1) ' legyen hiba, amelynek kódja 1!
    End If
    ' itt van valami - 3

Kilepes:
    ' hibafigyelés kikapcsolása:
    On Error GoTo 0 ' ha lesz később hiba, ide nem kerül a vezérlés!!!
    Exit Sub ' kilépés a szubrutinból
```

```
Hibak:
    If Err.Number = 1 Then ' hibás a db paraméter
      MsgBox ("Nem jó a db: '" & db & "'!")
      GoTo Kilepes
    End If

    ' egyéb hiba = hibás cím
    MsgBox ("Nem jó a tartomány cím: '" & hovaS & "'!")
    GoTo Kilepes
End Sub ' Eljárás
```

Feladat:

Emelje át a kódot a kódmodulba! Írjon az `' itt van valami - 1` helyére egy `hovaS`-re vonatkozó értékadást, először rossz címhivatkozást (pl. „`D%`”), aztán jót is (pl. „`D5`”). Járjon el hasonlóan a `' itt van valami - 2` hellyel is, de itt a `db` `0`-ságát tegye próbára. Minden esetben kövesse nyomon a futást!

Vajon miért nem hagyható el az **`End Sub`** előtti **`GoTo`** `Kilepes`?[7]

Hogy minimalizáljuk a kódolási hibákat, érdemes a program elejére beilleszteni az **`Option Explicit`** direktívát, amellyel „kierőszakoljuk” a változó deklarációkat. (Így elkerüljük a BASIC „klasszikus rossz szokását”, hogy egy változót az első előfordulásakor automatikusan és szó nélkül létrehozza, pedig lehet, hogy csak egy egyébként már létező, másik változó nevét írtuk el.)

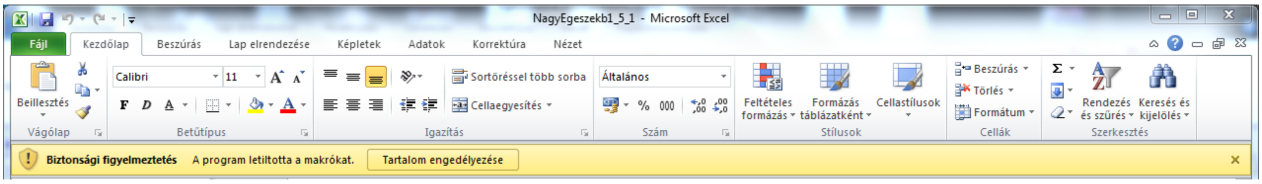
Feladat:

A korábbi feladatban szereplő „cellamásoló” függvényt tegye biztonságosabbá úgy, hogy ha a paramétertartomány nem egy cellára vonatkozik, akkor egy „`#ÉRTÉK!`” hibával térjen vissza![8]

S végül …

Ne feledjük, hogy egy VBA program akár vírus is lehet. Ezért e tartalmak kódban maradását és majdani aktivizálását explicit módon engedélyezni is kell az Excelben. Ez az oka, hogy egy speciális formátumban kell a fájlt kimenteni, amelynek kiterjesztése: **`xlsm`**.

Az újbóli megnyitáskor az Excel rákérdez, hogy a VBA program aktivizálható-e:



Biztonsági figyelmeztetés A program letiltotta a makrókat.

Ezt persze a saját táblázatunk esetében bátran engedélyezhetjük.

---

[7] Mert beállítva marad a hibacím… azaz ha később futási hiba következik be bárhol, akkor is ide kerül (teljességgel értelmetlenül!) a vezérlés!

[8] Ötlet: hibavizsgálat után, hibára az `Error (0)` utasítást adjuk ki, amelynek éppen a kívánt hibaállapot lesz a hatása.

4. A NAGYPONTOSSÁGÚ MŰVELETEK

A műveletek szignatúrája

Sajnos a VB-ben operátorok nem „definiálhatók felül”, vagyis –bármennyire is– elegáns lenne, nem fogjuk tudni a szokásos operátor-jeleket (+, –, *, /, < stb.) a nagypontosságú adatokra alkalmazni: függvényekként fogjuk definiálni. Továbbá, mivel a nagy-egészek „alakilag” **`String`**-ek, így az Excel nem is lenne képes felismerni, hogy egy speciálisan kezelendő **`String`**-ről van szó.

Egy **névkonvenciót** fogunk követni, amely megkönnyíti az alprogramok nevének észben tartását. Ha egy „fő” függvényről vagy eljárásról van szó, tehát amely elsődleges funkcióval bír: „**NE**”-vel fog kezdődni, utalva a „**N**agy-**E**gész’-ekkel való kapcsolatra. Mivel távlatilag többféle ábrázolásra is gondolunk, a későbbiektől való megkülönböztetés kedvéért a funkciót kifejező név mögé, utolsó jelként az „**S**” betűt rakjuk, amely a **`String`**-es ábrázolást fejezi ki. (Lehetne például rekordként definiálni.)

Próbálja kitalálni a nevük alapján az alábbi függvények és eljárások feladatát!

Relációk:

```
Function NENagyobbES(mi As String, minel As String) As Boolean
Function NEEgyenlőES(mi As String, mivel As String) As Boolean
```

Aritmetikaiak:

```
Function NENövelS(egesz As String) As String
Function NECsökkentS(egesz As String) As String
Function NEÖsszegS(op1 As String, op2 As String) As String
Function NEKülönbségS(op1 As String, op2 As String) As String
Function NEUnárisMínuszS(op As String) As String
Function NESzorzatS(op1 As String, op2 As String) As String
Function NEOsztásS(op1 As String, op2 As String) As String
Function NEDivS(op1 As String, op2 As String) As String
Function NEModS(op1 As String, op2 As String) As String
Function NELnkoS(op1 As String, op2 As String) As String
```

Egyebek:

```
Function NESgnS(egesz As String) As Integer
Function NEAbsS(egesz As String) As String
```

Feladat:

Magyarázza el, miért nem lehet a két relációt (**<**, **=**) sem egy az egyben felhasználni, azaz miért kell azokat is függvényként újradefiniálni!

---

[10] Az `InStr` és `InStrRev` függvény 0-t ad vissza, ha a keresett érték nem található meg.

A műveletek algoritmusa

Megállapodunk abban, hogy kétváltozós függvények esetében **csak azonos alapú** operandusokkal törődünk. Első közelítésben ezt azonban nem vizsgáljuk. Bár a hiba jelzése egyetlen **`If`** elegendő lenne:

```
If a1 <> a2 Then Error (0)
```

Itt az `a1` és az `a2` az operandusok alapját tartalmazó változókat jelöli. Feltételeztük, hogy a `SzétszedS` (korábban emlegetett) eljárással már darabjaira szedtük az operandusokat.

Azaz ha az `a1` és az `a2` alapok különböznek, akkor `0` hibakóddal a végrehajtás megszakad. Az Excelbe visszatérve az „`#ÉRTÉK!`” hibaüzenet jelenik meg.

Továbbá feltesszük, hogy a **paraméterek szintaxisa helyes**, némi engedménnyel. Amiatt, hogy az Excel táblába a fentebb definiált alakú nagy-egészeket be lehessen írni, megengedjük, hogy szóközzel kezdődhessen. Például a helyes „`-123456789ABCDEF(16)`” helyett a (szóközzel kezdődő) „ `-123456789ABCDEF(16)`”-t is jónak fogadjuk el. A nyitó szóközök eliminálása könnyen megoldható. Például a `mi`-ben tárolt paraméterszövegnek előfeldolgozása VB-ben ennyi:

```
mi = Trim(mi) ' a szóközök eltávolítása                  (SzóközLenyelés)
```

Sok gond forrása lehet a nem sok értelemmel bíró, bár aritmetikailag helyes **nulla előtti negatív** előjel, illetve a **vezető nullák** esete. Ezért a paraméter átvétel után célszerű ezektől a zavaró tényezőktől megszabadulni. Hasznos egy olyan eljárást írni mindenekelőtt, amely –a fentiek figyelembe vételével– elemeire szed egy nagy-egészet. Ez a **`SzétszedS`** eljárás. Egy kezdetleges változatát mintaként mellékeljük. Paramétereiről: `egesz` – a bemeneti nagy-egész; kiemenetiek – az `alap` a bemeneti nagy-egész alapja, az `ej` az előjele, az `egész2` maga a számjegysorozat, a `h` az `egész2` hossza.

Feladat:

Sorról sorra értelmezze és magyarázza el az alábbi eljárás működését!

```
Sub SzétszedS(egesz As String, alap As Integer, ej As String, _
              h As Integer, egesz2 As String)
'
' Ef: 'egesz': szintaktikusan helyes nagy-egész, alakja: [+/-]J..J[(A..A)]
'               J=0..9 -- a számok jegyei, A=0..9 -- a számrendszer alapszáma
'     vezető szóközöket tartalmazhat, de azt elimináljuk; vezető 0-k nincsenek
'
    egesz = Trim(egesz) ' a szóközök eltávolítása
    ' az alap és a számjegysorozat szétválasztása:
    Dim holZaro As Integer, holNyito As Integer
    holNyito = InStr(egesz, "("): holZaro = InStrRev(egesz, ")") [10]
    If holNyito < holZaro Then
      alap = Val(Mid(egesz, holNyito + 1, Len(egesz) - holNyito - 1)) [11]
      egesz2 = Left(egesz, holNyito - 1): h = Len(egesz2)
    Else
      alap = 10
      egesz2 = egesz: h = Len(egesz2)
    End If
```

---

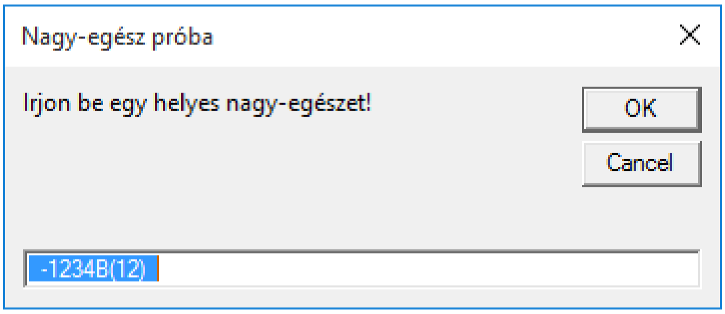
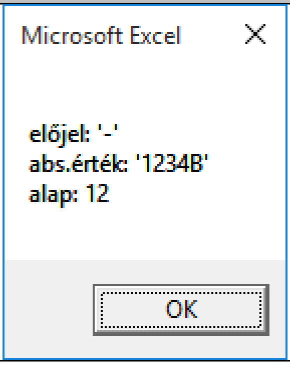
[10] Az `InStr` és `InStrRev` függvény 0-t ad vissza, ha a keresett érték nem található meg.

[11] A `Mid` „darab” paramétere –a szintaktikai helyesség miatt– így is meghatározható lenne: `holZáró−holNyito`.

```
    ' előjel:
    If Left(egesz2, 1) = "-" Then
      ej = "-": egesz2 = Right(egesz2, h - 1): h = h - 1
    ElseIf Left(egesz2, 1) = "+" Then
      ej = "": egesz2 = Right(egesz2, h - 1): h = h - 1
    Else
      ej = ""
    End If
    ' előjel-korrekció ("-0" kizárása):
    If ej = "-" And egesz2 = "0" Then ej = ""
End Sub ' SzétszedS
```

Feladat:

Emelje át kódmodulba az előbbi szubrutint, és készítsen egy ezt kipróbáló (mondjuk nevű) eljárást, amely bekér egy megengedett nagy-egészet és kiírja a kimenő paraméterek értékeit! Valahogy így:

Beviteli ablak	Eredmény-ablak
 Irjon be egy helyes nagy-egészet! -1234B(12)	 előjel: '-' abs.érték: '1234B' alap: 12

Vegye észre, az eredmény ablak többsoros! Az egyes sorok közé a sorvégjel karaktert (`Chr(13)` vagy `vbNewLine` előredefiniált konstans) kell illeszteni!

Feladat:

Hogyan gyengíthető az előfeltétel?

A kódmodulba átemelt szubrutint bővítse úgy, hogy az esetleges vezető 0-kat hagyja el!

Feladat:

A `SzétszedS`-re építve alkossa meg az alábbiakat! Amint elkészült egy függvénnyel, mindjárt próbálja is ki: írjon be az Excel számolótáblába egy-két, az adott függvényt tartalmazó hivatkozást!

```
Function NEEgyenlőES[12](mi As String, mivel As String) As Boolean
    a paraméterek előkészítése (SzóközLenyelés+ÁlElőjelLenyelés)
    a paraméterek szétszedése: előjelre (miEj, mivelEj) és értékes jegyekre
      (mi2, mivel2) … esetleg számrendszer-alapra

    NEEgyenlőES = miEj = mivelEj And mi2 = mivel2
End Function ' NEEgyenlőES
```

---

[12] A név további magyarázatául: ..`EgyenlőE`.. = egyenlő-e?

```
Function NENagyobbES(mi As String, minel As String) As Boolean
    a paraméterek előkészítése (SzóközLenyelés+ÁlElőjelLenyelés)
    a paraméterek szétszedése: előjelre (miEj, minelEj) és értékes jegyekre
      (mi2, minel2) … esetleg számrendszer-alapra

    az előjelek alapján eldönthető esetek
    azonos előjelek esetén:
      a lexikografikus rendezéshez a rövidebb kiegészítése vezető 0-kkal
      az értékes jegyek rendezettsége alapján a válasz megadása
End Function ' NENagyobbES

Function NENövelS(egesz As String) As String
    a paraméter előkészítése (SzóközLenyelés+ÁlElőjelLenyelés)
    a paraméterek szétszedése: előjelre (egeszEj) és értékes jegyekre
      (egesz2), számrendszer-alapra (alap)

    az értékes jegyekkel fogunk alább dolgozni
    Ha nem negatív a szám, akkor
      számjegyenként hátulról előre felé haladva átvitelt képezve meghatározzuk
      az eredmény jelsorozatot. Az átvitel kezdetben 1.
      Közben ügyelünk a számrendszer-alapra, és arra,
      hogy nem 0 átvitel maradhat a végén is!
    Ha negatív a szám akkor
      Ha egesz2=”1” akkor
        az eredmény triviálisan kész, hiszen „0”
      különben
        számjegyenként hátulról előre felé haladva átvitelt képezve meghatároz-
        zuk az eredmény jelsorozatot. Az átvitel kezdetben -1.
        Közben ügyelünk a számrendszer-alapra, és arra,
        hogy nem 0 átvitel maradhat a végén is!
    NENövelS = összeállítjuk a teljes nagy-egészet
End Function ' NENövelS

Function NECsökkentS(egesz As String) As String
    a paraméter előkészítése (SzóközLenyelés+ÁlElőjelLenyelés)
    a paraméterek szétszedése: előjelre (egeszEj) és értékes jegyekre
      (egesz2), számrendszer-alapra (alap)

    az értékes jegyekkel fogunk alább dolgozni
    Ha nem negatív a szám, akkor
      Ha egesz2=”1” akkor
        az eredmény triviálisan kész, hiszen „0”
      különben
        számjegyenként hátulról előre felé haladva átvitelt képezve meghatároz-
        zuk az eredmény jelsorozatot. Az átvitel kezdetben -1.
        Közben ügyelünk a számrendszer-alapra, és arra,
        hogy nem 0 átvitel maradhat a végén is!
    Ha negatív a szám akkor (felhasználhatjuk az NENövelésS-t)
      számjegyenként hátulról előre felé haladva átvitelt képezve meghatározzuk
      az eredmény jelsorozatot. Az átvitel kezdetben 1.
      Közben ügyelünk a számrendszer-alapra, és arra,
      hogy nem 0 átvitel maradhat a végén is!
    NECsökkentS = összeállítjuk a teljes nagy-egészet
End Function ' NECsökkentS

Function NEÖsszegS(op1 As String, op2 As String) As String
    a paraméter előkészítése (SzóközLenyelés+ÁlElőjelLenyelés)
    a paraméterek szétszedése: előjelre (ej1, ej2) és értékes jegyekre
      (e12, e22), számrendszer-alapra (alap)

    a művelet egyszerűsödik, ha az operandusok számjegy-sorozatai (e12,e22)
    azonos hosszúak, tehát a rövidebbet egészítsük ki vezető 0-kkal
```

```
    Ha azonos előjelűek akkor
      számjegyenként hátulról előre felé haladva átvitelt képezve összeadással
      meghatározzuk az eredmény jelsorozatot. Az átvitel kezdetben 0.
      Közben ügyelünk a számrendszer-alapra, és arra,
      hogy nem 0 átvitel maradhat a végén is!
    Ha különböző előjelűek, akkor
      a nagyobb abszolút értékűből vonjuk ki a kisebbiket;
      azaz esetleg (e12,e22) csere
      képezzük az eredmény előjelét;
      számjegyenként hátulról előre felé haladva átvitelt képezve meghatározzuk
      kivonással az eredmény jelsorozatot. Az átvitel kezdetben 0.
      Közben ügyelünk a számrendszer-alapra, és arra,
      hogy nem 0 átvitel maradhat a végén is,
      és vezető 0-k keletkezhetnek, amit törölni kell!
    NEÖsszegS = összeállítjuk a teljes nagy-egészet
End Function ' NEÖsszegS

Function NEUnárisMínuszS(op As String) As String
    a paraméter előkészítése (SzóközLenyelés+ÁlElőjelLenyelés)

    az első jel alapján triviálisan megadható a válasz
End Function ' NEUnárisMínuszS
```

Visszavezetjük az összeadás és az unáris mínusz műveletekre:

```
Function NEKülönbségS(op1 As String, op2 As String) As String
    Dim op2M As String, kulonbseg As String
    op2M = NEUnárisMínuszS(op2)
    NEKülönbségS = NEÖsszegS(op1, op2M)
End Function ' NEKülönbségS
```

A szorzás algoritmusát, pontosabban kódját mellékeljük. Ötleteket is vehet innen az előzőek némelyikéhez.

```
Function NESzorzatS(op1 As String, op2 As String) As String
'
' Ef: op1, op2 szintaktikusan helyes nagy-egészek: [+/-]J..J[(A..A)]
'     J=0..9 -- a számok jegyei, A=0..9 -- a számrendszer alapszáma
'     továbba azonos alapúak
'     ha az Ef nem teljesül, akkor "ERROR 0"
'     vezető szóközöket tartalmazhat, de azt elimináljuk
'
    ' kimenet:
    Dim szor As String, a As Integer, ej As String, h As Integer, _
        szor2 As String
    ' bemenet:
    Dim e1 As String, e2 As String, a1 As Integer, a2 As Integer
    Dim ej1 As String, ej2 As String, h1 As Integer, h2 As Integer
    Dim e12 As String, e22 As String
    Call SzétszedS(op1, a1, ej1, h1, e12)
    Call SzétszedS(op2, a2, ej2, h2, e22)
    If a1 <> a2 Then Error (0)
    ' szorzás:
    Dim jegy1 As Integer, jegy2 As Integer, atv As Integer
    Dim rjegy1 As Integer, rjegy2 As Integer
    Dim rszor As String ' részletszorzat = a szorzás egy "sora"
    Dim rh As Integer ' a részletszorzat hossza
    Dim lepes As String ' a részletszorzatok léptetéséhez
    Dim i As Integer, j As Integer
    Dim szor2tmp As String ' a szor2 "párja"
```

```
    a = a1
    szor2 = "0"
    If ej1 = ej2 Then
      ej = ""
    Else ' különböző előjelűek
      ej = "-"
    End If
    lepes = ""
    For i = h2 To 1 Step -1 ' az op2 jegyein haladunk, hátulról
      atv = 0: rszor = ""
      jegy2 = ValSzámjegy(Mid(e22, i, 1)) ' a számjegy értéke
      For j = h1 To 1 Step -1 ' az op1 jegyein haladunk, hátulról
        jegy1 = ValSzámjegy(Mid(e12, j, 1))
        rszor = Számjegy(Trim(Str((jegy1 * jegy2 + atv) Mod a))) & rszor
        atv = Int((jegy1 * jegy2 + atv) / a)
      Next j
      If atv > 0 Then rszor = Trim(Str(atv)) & rszor
      ' szor2-ben az eddigi szorzatok összege
      ' rszor-ban az aktuális op2 számjeggyel képzett szorzat
      ' rszor léptetése "balra" i-1 jeggyel:
      rszor = rszor & lepes
      lepes = lepes & "0"
      ' szor2-t megfelelő hosszúra kiegészítjük:
      h = Len(szor2): rh = Len(rszor)
      szor2 = Sok0(rh - h) & szor2
      ' hozzáadjuk a részlet szorzatot a szorzathoz:
      szor2tmp = "" ' ebben képződik az összeg
      atv = 0
      For j = rh To 1 Step -1 ' az op1 jegyein haladunk, hátulról
        rjegy1 = ValSzámjegy(Mid(szor2, j, 1))
        rjegy2 = ValSzámjegy(Mid(rszor, j, 1))
        szor2tmp = Számjegy(Trim(Str((rjegy1 + rjegy2 + atv) Mod a))) & szor2tmp
        atv = Int((rjegy1 + rjegy2 + atv) / a)
      Next j
      If atv = 1 Then szor2tmp = "1" & szor2tmp
      szor2 = szor2tmp
    Next i
    ' az eredmény összeállítása:
    If a = 10 Then szor = ej & szor2 Else szor = ej & szor2 & "(" & a & ")"
    NESzorzatS = szor
End Function ' NESzorzatS
```

Az osztás függvény különlegessége, hogy egyetlen szövegbe kell zsúfolnia két nagy-egészet: a hányados egészrészét (`TDIV`) és a maradékot (`TMOD`). Ehhez egy sémát kell definiálnunk. Legyen a következő az eredmény szintaxisa:

`TDivMod = TDiv & "|" & TMod` (TDivMod)

`TDiv = TElőjel & TSzámjegyek & TSzámrendszer` (TDiv)

`TMod = TSzámjegyek & TSzámrendszer` (TMod)

A nevezőből elspóroljuk az előjelet, hiszen az –definíció szerint– úgy sem lehet negatív, és a "+" előjel elhagyható.

Érdemes egy kicsit előre gondolni: ugyanilyen módon lehetne a nagypontosságú racionális számokat is egyetlen szöveggel leírni. Például így:

`TRac = TSzámláló & "/" & TNevező` (TRac1)

vagy

```
      TRac = TEgészrész & "," & TTörtrész                                    (TRac2)
```

Az osztás szokásos, „kézi algoritmusát” valósítottuk meg alább. A kódját mellékeljük.

```
unction NEOsztásS(op1 As String, op2 As String) As String
'
' Ef: op1, op2 szintaktikusan helyes nagy-egészek: [+/-]J..J[(A..A)]
'     J=0..9 -- a számok jegyei, A=0..9 -- a számrendszer alapszáma
'     továbba azonos alapúak
'     és  op2<>0
'     ha az Ef nem teljesül, akkor "ERROR 0"
'     vezető szóközöket tartalmazhat, de azt elimináljuk
' Uf: NEOsztásS(x,y)=d&"|"&m
'     ahol d és m nagy-egészek, d=egész hányados, m=maradék; m>=0
'
    ' kimenet:
    Dim d As String, m As String
    Dim d2 As String, m2 As String, a As Integer, ej As String
    ' bemenet:
    Dim e1 As String, e2 As String, a1 As Integer, a2 As Integer
    Dim ej1 As String, ej2 As String, h1 As Integer, h2 As Integer
    Dim e12 As String, e22 As String
    Call SzétszedS(op1, a1, ej1, h1, e12): Call SzétszedS(op2, a2, ej2, h2, e22)
    If a1 <> a2 Or e22 = "0" Then Error (0)
    ' osztás:
    a = a1
    If ej1 = ej2 Then
      ej = ""
    Else ' különböző előjelűek
      ej = "-"
    End If
    Dim szam As String, nev As String ' "számláló", "nevező"
    Dim még0 As Boolean, rd As Integer ' segéd változók
    szam = e12: nev = e22
    If NENagyobbES(nev, szam) Then ' kész az eredmény
      m2 = szam: d2 = "0"
    Else ' még számolni kell!
      d2 = "": még0 = True ' a hányados, ami még 0
      m2 = "" ' a (mindenkori) maradék
      While szam <> ""
        ' a vezető 0-k elhagyása mellett a maradék köv. számjeggyel bővítése:
        If m2 = "0" Then m2 = Left(szam, 1) Else m2 = m2 & Left(szam, 1)
        ' a még számláló megmaradó számjegyei:
        szam = Right(szam, Len(szam) - 1)
        rd = 0 ' rd=INT(számláló / nevező), 'a' alapú számrendszer esetében!!!
        While NENagyobbES(m2, nev) Or NEEgyenlőES(m2, nev)
          m2 = NEAlappalS(m2, a): nev = NEAlappalS(nev, a) ' szám-alap kell
          m2 = NEKülönbségS(m2, nev): rd = rd + 1
          m2 = NEAlapNélkülS(m2): nev = NEAlapNélkülS(nev)
        Wend
        If még0 Then
          d2 = Számjegy(rd) ' a számrendszer figyelembe vétele
          még0 = rd = "0"
        Else
          d2 = d2 & Számjegy(rd) ' a számrendszer figyelembe vétele
        End If
      Wend
    End If
    ' az eredmény összeállítása:
    If Len(m2) > 1 And Left(m2, 1) = "0" Then _
      m2 = Right(m2, Len(m2) - 1) ' vezető 0 levágása
```

```
    If a = 10 Then
      If d2 = "0" Then d = d2 Else d = ej & d2
      m = m2
    Else ' nem decimális
      If d2 = "0" Then d = d2 & "(" & a & ")" Else d = ej & d2 & "(" & a & ")"
      m = m2 & "(" & a & ")"
    End If
    NEOsztásS = d & DivModJel & m
End Function ' NEOsztásS
```

Az osztás fenti függvényének eredményéből igen egyszerűen megkapható az alábbi két függvény eredménye:

```
Function NEDivS(op1 As String, op2 As String) As String
    …
End Function ' NEDivS

Function NEModS(op1 As String, op2 As String) As String
    …
End Function ' NEModS
```

A legnagyobb közös osztó keresésének klasszikus algoritmusa a jólismert Euklideszi algoritmus:

```
Function NELnkoS(op1 As String, op2 As String) As String
'
' Ef: op1, op2 szintaktikusan helyes nagy-egészek: [+/-]J..J[(A..A)]
'     J=0..9 -- a számok jegyei, A=0..9 -- a számrendszer alapszáma
'     továbba azonos alapúak
'     és  op2<>0
'     ha az Ef nem teljesül, akkor "ERROR 0"
'     vezető szóközöket tartalmazhat, de azt elimináljuk
' Uf: NELnkoS(x,y)=Lnko(op1,op2)
'
    ' bemenet:
    Dim e1 As String, e2 As String, a1 As Integer, a2 As Integer
    Dim ej1 As String, ej2 As String, h1 As Integer, h2 As Integer
    Dim e12 As String, e22 As String
    Call SzetszedS(op1, a1, ej1, h1, e12): Call SzetszedS(op2, a2, ej2, h2, e22)
    If a1 <> a2 Or e22 = "0" Then Error (0)
    ' Lnko-képzés:
    Dim a As String, b As String, r As String
    a = NEAbsS(op1): b = NEAbsS(op2)
    If NENagyobbES(b, a) And NESgnS(a) <> 0 Then
      r = a: a = b: b = r ' a<->b
    End If
    r = NEModS(a, b)
    While NESgnS(r) <> 0
      a = b: b = r: r = NEModS(a, b)
    Wend
    NELnkoS = b
 End Function ' NELnkoS
```

További hasznos eljárás és függvény

Többször kellett sok-sok 0-val kiegészíteni egy nagy-egészet. Az alábbi függvény ehhez jól jöhet:

```
Function Sok0(db As Integer) As String
'
' Uf: Sok0(db)=db darab "0" jel
'
```

```
    Dim i As Integer, nullak As String
    nullak = ""
    For i = 1 To db
      nullak = nullak & "0"
    Next i
    Sok0 = nullak
End Function ' Sok0
```

A számrendszerek figyelembe vételét az alábbi két függvény egységesen és könnyedén végzi el. A `Számjegy` nevű a szokásokat figyelembe vevő jelekké transzformálja az értékeket. `0..9` → `"0".."9"`, `10..35` → `"A".."Z"`, azért eddig, mert kézenfekvő jelek utolsója a `"Z"`. Innen adódik a számrendszer-alapra a 36-os korlát.

```
Function Számjegy(i As Integer) As String
'
' Uf: Számjegy(i)=a legfeljebb 36-alapú számrendszer i értékű számjegye
'
    If i < 10 Then Számjegy = Chr(i + 48) Else Számjegy = Chr(i - 10 + 65)
End Function ' Számjegy
```

Az előbbi inverze a `ValSzámjegy`[13] függvény, amely egy –megengedett– jelhez generálja az értékét.

```
Function ValSzámjegy(s As String) As Integer
'
' Ef: s megengedett karakter (s ELEME ['0'..'9', 'A'.. 'Z'])
' Uf: ValSzámjegy(s)=a legfeljebb 36-alapú számrendszer s jelének az értéke
'
    If s <= "9" Then ValSzamjegy = Val(s) Else ValSzámjegy = Asc(s) + 10 - 65
End Function ' ValSzámjegy
```

A modult letöltheti innen:

http://people.inf.elte.hu/szlavi/InfoRendsz/Nagyarit/Aritmetika/NEModul.bas

Ezt kell az Excel VBA moduljaként importálni.

---

[13] A függvény elnevezés magyarázatául: a `Val` „előke” utalás arra, hogy a VB beépített `Val` függvényének egyfajta **kiterjesztése**. (Persze egyben leszűkítése is, hiszen csak **egyetlen jel** esetén alkalmazható!)

5. KIPRÓBÁLÁS

A fentiek kipróbálását célszerűen úgy végezheti, hogy a táblázatban változatos bemenetekkel kiszámoltatja a legkülönfélébb függvényeket.
De úgy is, hogy például megalkot néhány érdekes függvényt a VBA-ban, amely intenzíven megmozgatja az összeadás és a szorzás függvényeket. Érdemes párhuzamosan az Excel saját megoldásaival egybe vetni. Érdekes kérdés: meddig tudja követni az Excel a mi számítási pontosságunkat?

1. házi feladat

A Fibonacci sorozat – rekurzív és iteratív implementáció

2. házi feladat

A faktoriális sorozat – rekurzív és iteratív implementáció

3. házi feladat

A binomiális együtthatók: $\binom{n}{k}$ – rekurzív és iteratív implementáció

4. házi feladat

Az Ackermann-féle függvénynek [14] sok különlegessége van. Az egyik: mindkét paramétere (eltérően az előzőektől) maga is nagy-egész kell legyen!

---

[14] L. Szlávi–Zsakó: „*Módszeres programozás – Rekurzió*” (μlógia 4) vagy az „Algoritmizálás, adatmodellezés tanítása” e-anyag 2.3. fejezetében: http://algtan1.elte.hu/downloads/scorm/lecke1_lap2.html#hiv5 .