EXCEL VBA FÜGGVÉNY KIALAKÍTÁSA FONTOSABB IRRACIONÁLIS SZÁMOK FELISMERÉSÉRE

Fabulya Zoltán

Absztrakt: Számítások során sokszor kapunk eredményként irracionális értéket, melynek megjelenítése tizedes tört alakban látható számológépünk vagy számítógépünk kijelzőjén. Viszont így nehezen értelmezhető az érték és pontatlan is. Ezért készítettünk olyan Excel függvényt, hogy a fontosabb irracionális értéken alapuló számokat, mint egy egész szám gyökeként vagy a Pi szám racionális törttel szorzásakor adódó irracionális számokat, meg tudjuk jeleníteni a „√” (gyök) vagy a „π” (Pi) karakterek segítségével. A programozás során kifejlesztett algoritmusok a Microsoft Excel táblázatkezelő program Visual Basic for Applications szolgáltatásával készültek el. A szám megfelelő megjelenítéséhez a függvényünknek fel kell ismernie az argumentumaként kapott számot, melyet szöveges formátumban jelenít meg rövidebb, és jobban értelmezhető formában, mint az eredeti, tizedes tört alakú irracionális szám.

Abstract: During calculations, we often get an irrational value, the display of which is shown as a decimal fraction on the display of our calculator or computer. However, the value is difficult to interpret and inaccurate. Therefore, we created an Excel function so that numbers based on a major irrational value, such as irrational numbers resulting from the root of an integer or the number of Pi multiplied by a rational fraction, can be represented by the characters ”√” (root) or ”π” (Pi). The algorithms developed during programming were developed using the Visual Basic for Applications feature of the Microsoft Excel spreadsheet program. To display the number properly, our function must recognize the number obtained as an argument, which it displays in text format in a shorter and more interpretable form than the original, decimal fractional irrational number.

Kulcsszavak: irracionális szám, Excel VBA, programozás, függvény, gyök felismerése

Keywords: irrational number, Excel VBA, programming, function, root recognition

1. Bevezetés

Az irracionális számok tizedes tört alakban jelennek meg elektronikus eszközeink kijelzőjén. Szükségszerűen így a pontos érték nem lesz kiolvasható, mert végtelen számú tizedes számjegy alkotja az irracionális számokat, melyben a végtelenségig ismétlődő szakasz sincs (Obádovics, 2012). Ilyen a Pi (π) szám is, de gyökvonás eredményeként szintén irracionális szám adódik még az egész számok esetén is, ha az nem négyzetszám. Rövidebb, jobban értelmezhető és pontos az irracionális szám, ha a megfelelő szimbólumokat, például a gyök (√) vagy a Pi (π) karaktereket alkalmazzuk a leírásuk során. Ezt a fajta megjelenítést viszont a programjaink nem támogatják.

A szögek megadása sokszor radián mértékegységben szükséges. Egy nevezetes szög tizedes tört alakja radiánban szinte értelmezhetetlen, mint ahogy ez látható az (*1*) képletben.

$$30° = \frac{\pi}{6} = 0{,}523598775598299 \qquad (1)$$

ISSN 1788-7593 (Nyomtatott) DOI: https://doi.org/10.14232/jtgf.2021.3-4.77-86
ISSN 2676-9867 (Online)

A trigonometrikus függvények értékei nevezetes szögek esetében könnyebben értelmezhetők gyökös formában (*2*).

$$\cos 30° = \frac{\sqrt{3}}{2} = 0{,}866025403784439 \tag{2}$$

A fenti képletekben egyenletszerkesztő biztosította a megfelelő megjelenítést, de karakteres mód is alkalmazható (π/6, √3/2). Ez a mód már lehetővé teszi, hogy az Excel táblázatkezelő programban karakteresen, szöveg típusúan kiolvasható legyen egy számítás eredményeként kapott, tizedes tört alakú szám. Ebben a cikkben bemutatásra kerül a legfontosabb típusú irracionális számok karakteres módú kijelzéséhez szükséges Excel munkalapfüggvény kialakítása, melynek lényeges eleme, a számok felismerését biztosító algoritmus kifejlesztése a Visual Basic for Applications (VBA) által nyújtott fejlesztői környezetben. Számos cikk kutatása használja az Excel VBA lehetőségeit (Hampel, 2017; Hampel, 2018).

Mivel végtelen sok irracionális szám van, s köztük olyanok is, melyek nem jeleníthetők meg speciális karakterek segítségével, ezért csak néhány, fontosabb típusú irracionális szám felismerésével foglalkozunk, melyek a következő alakúak (*3*) és (*4*).

$$\frac{m}{n} \cdot \pi \tag{3}$$

$$\frac{m}{n} \cdot \sqrt{a} \tag{4}$$

ahol:
$m =$ egész szám
$n =$ pozitív egész szám $(1 \leq n)$
$a =$ pozitív egész szám $(1 \leq a)$

Tehát vagy a Pi számon, vagy egy egész szám négyzetgyökén alapul a felismerendő irracionális szám. Az *n* és *a* számok esetén további korlátozás szükséges, hiszen a végtelenségig nem terjeszthetjük ki a felismerést. A tesztek során a 100 felső korlát elegendőnek bizonyult. Azért szükséges, hogy *m* és *n* egész számok legyenek, hogy a megjelenítési problémát okozó irracionális érték csak a Pi vagy a gyökvonás eredménye lehessen.

2. Anyag és módszer

A kutatásunk elméleti háttere az irracionális számok ismeretkörébe tartozik, míg a programozási feladatokhoz az Excel VBA áttekintése szükséges.

2.1. Irracionális számok

Definíció szerint, az irracionális számok azok a valós számok, melyek nem írhatók fel két egész szám hányadosaként. Ismeretes, hogy a π és a $\sqrt{2}$ irracionális számok.

Könnyen bizonyítható, hogy a pozitív egész számok négyzetgyöke is irracionális, kivéve a négyzetszámokat, melyek négyzetgyöke egész szám (Obádovics, 2012).

Az irracionális számok végtelen számú tizedesjegyből állnak, ismétlődő szakaszok nélkül. Emiatt nem írható le a pontos értékük tizedes tört alakban, de tetszőleges pontosság elérhető a tizedesjegyek számának növelésével. Így viszont hosszabb lesz a leírt szám.

Az irracionális számok egy fontos csoportját azok képezik, melyek egy racionális szám négyzetgyökeként adódnak. Egy másik csoportjuk azok, melyek radián mértékegységben egy olyan szöget jelölnek, melyek racionális értékkel adhatók meg fok mértékegység esetén. Mint később látni fogjuk, ezek az irracionális számok felírhatók a (*3*) vagy a (*4*) képlettel megadott formában. Célunk olyan Excel VBA függvény kifejlesztése, mellyel megjeleníthető ilyen alakban egy irracionális szám, ha ez egyáltalán lehetséges.

2.2. Az Excel VBA

Az Excel táblázatkezelő program használata során jellemzően számított eredményekre van szükségünk. Ekkor formulákat és munkalapfüggvényeket alkalmazunk. Viszont arra is van lehetőségünk, hogy saját függvényeket hozzunk létre (Kovalcsik, 2005). Ehhez programoznunk kell a Visual Basic for Applications szolgáltatást kihasználva, mely a Basic programozási nyelven alapul. A függvények névvel rendelkeznek, és az argumentumukban szereplő változóktól függő eredményt adnak (Matteson, 1995). Általános alakjuk:

```
Public Function függvénynév(argumentumok)
  utasítások
  függvénynév = érték
End Function
```

A függvények használatakor meg kell adnunk az argumentumuk értékét. Az utasítások végrehajtásával kialakul egy érték, mely a függvény eredménye lesz a `függvénynév = érték` utasítás miatt.

A függvényben szereplő utasítások alapesetben egyszer, egymás után hajtódnak végre. Ezt nevezzük szekvenciális utasítás szerkezetnek, melynek a következő az alakja (Kovalcsik, 2005):

```
…
utasítás1
utasítás2
utasítás3
…
```

Szelekciós szerkezetre van szükségünk akkor, amikor feltételtől függ az utasítás végrehajtása (Kovalcsik, 2005):

```
If feltétel Then
  utasítások1
Else
  utasítások2
End If
```

Ekkor a feltétel teljesülésekor az `utasítások1` blokk utasításai hajtódnak végre, ellenkező esetben pedig az `utasítások2` blokké.

Iterációs szerkezetre van szükségünk a programunkban akkor, amikor többször szükséges végrehajtani egy utasítást (Kovalcsik, 2005). Az alábbi feltételes ciklus szerkezetben a `feltétel` teljesüléséig az `utasítások` ismételt végrehajtása történik meg:

```
While feltétel
  utasítások
Wend
```

A fent leírt szekvenciális, szelekciós és iterációs szerkezetekkel bármely program megvalósítható.

2.3. Kereső algoritmus

Munkánk lényeges pillére, hogy egy számról felismerjük azt, hogy előállítható-e a számunkra szükséges formában. Ezt a feladatot egy kereső algoritmussal oldhatjuk meg. Ennek lényege, hogy a rendelkezésre álló elemek listájából kikeresi azt, mely megfelel a feltételeknek, ha egyáltalán van ilyen közöttük. Ezt úgy végezhetjük el, hogy az elemeket egyenként vizsgáljuk mindaddig, míg vagy el nem fogynak, vagy nem találunk megfelelőt (Walkenbach, 2013). Ez megvalósítható egy feltételes ciklus szerkezettel. A ciklus befejezésekor még egy szelekciós szerkezettel vizsgáljuk meg, hogy a ciklus sikeres keresés miatt fejeződött be, vagy azért, mert elfogytak a lista elemei. Az algoritmus pszeudokódja az alábbi:

```
Legyen a vizsgálandó elem a lista első eleme
While a vizsgálandó elem nem megfelelő és _
                              van még nem vizsgált elem
  Legyen a vizsgálandó elem a lista következő eleme
Wend
If a vizsgálandó elem megfelelő Then
  A találat a vizsgálandó elem
Else
  Nincs találat
End If
```

3. Eredmények

A kutató munka során meg kell vizsgálnunk, hogy milyen feltételek mellett mondhatjuk egy számról, hogy a (*3*) vagy (*4*) formában felírhatók. Erre azért van szükség, hogy lehetőleg ne korlátozzuk az adott formában megjeleníthető számok körét. A további feladatunk a szükséges algoritmusok programjának elkészítése, melynek utolsó fázisa annak az Excel munkalapon használható függvénynek az elkészítése, mely a szám megjelenítését eredményezi az adott (*3*) vagy (*4*) formátumban.

3.1. A pontosan megjeleníthető irracionális számok

Rövid és pontos csak úgy lehet egy irracionális szám megadása, ha kifejezhető véges jelsorozattal leírható, ismert értékű irracionális számként, vagy annak véges jelsorozattal leírható függvényeként. A (*3*) vagy (*4*) forma esetén a Pi vagy a gyök jel szükséges ehhez.

Elsőként azt látjuk be, hogy bármely olyan szögérték, mely fok mértékegységben racionális számmal $\left(\frac{p}{q}\right)$ megadható, az felírható a (*3*) formában, ahogy ez az (*5*) levezetésben látható.

$$\frac{p}{q}[°] = \frac{p}{q} \cdot \frac{\pi}{180}[rad] = \frac{p}{180 \cdot q} \cdot \pi[rad] = \frac{m}{n} \cdot \pi[rad] \tag{5}$$

ahol:
$p =$ egész szám
$q =$ pozitív egész szám $(1 \leq q)$
$m =$ egész szám $(m = p)$
$n =$ pozitív egész szám $(n = 180 \cdot q)$

A (*6*) levezetésben az látható, hogy nem csak egy nem negatív egész szám gyöke, de bármely nem negatív racionális szám gyöke is felírható a (*4*) formában.

$$\sqrt{\frac{p}{q}} = \sqrt{\frac{p \cdot q}{q^2}} = \frac{\sqrt{p \cdot q}}{q} = \frac{1}{q} \cdot \sqrt{p \cdot q} = \frac{m}{n} \cdot \sqrt{a} \tag{6}$$

ahol:
$p =$ nem negatív egész szám
$q =$ pozitív egész szám $(1 \leq q)$
$m =$ egész szám $(m = 1)$
$n =$ pozitív egész szám $(n = q)$
$a =$ pozitív egész szám $(1 \leq n = p \cdot q)$

A következő (*7*) levezetésben arra látunk példát, hogy egy egész szám gyöke akár többféle (*4*) alakban is megadható, kisebb egész szám gyökének segítségével.

$$\begin{aligned} \sqrt{32} &= \sqrt{4 \cdot 8} = 2 \cdot \sqrt{8} \\ \sqrt{32} &= \sqrt{16 \cdot 2} = 4 \cdot \sqrt{2} \end{aligned} \tag{7}$$

Olyan kereső algoritmust fejlesztünk ki, mely a legkisebb pozitív egész szám gyökeként biztosítja a (*4*) formát.

3.2. Az elkészített függvények

Elsőként egy olyan univerzális kereső függvényt készítünk el, mely a (*3*) és a (*4*) alakok esetében is használható. Erre azért van lehetőségünk, mert ezek az alakok általánosíthatók, ahogy ezt a (*8*) képlet mutatja.

$$\frac{m}{n} \cdot \pi = \frac{m}{n} \cdot y \qquad (8)$$

$$\frac{m}{n} \cdot \sqrt{a} = \frac{m}{n} \cdot y$$

ahol:
$m =$ egész szám
$n =$ pozitív egész szám $(1 \leq n)$
$a =$ pozitív egész szám $(1 \leq a)$
$y =$ általánosított érték ($y = \pi$ vagy $y = \sqrt{a}$)

A kereső algoritmusnak akkor kell sikeres keresést jeleznie, ha a felismerendő szám (x) előállítható a (*8*) alakban, azaz teljesül a (*9*) összefüggés.

$$\mathrm{x} = \frac{m}{n} \cdot y \qquad (9)$$

ahol:
$x =$ a felismerendő szám
$m =$ egész szám
$n =$ pozitív egész szám $(1 \leq n)$
$y =$ általánosított érték

További érdekessége ennek az általánosításnak, hogy $y = 1$ esetén a felismerendő számot két egész szám hányadosaként állíthatjuk elő.

A kereső függvényünk kétváltozós, x és y függvényében keres olyan n pozitív egész számot, melyre a (*9*) összefüggés teljesül valamilyen m egész szám esetén. A (*9*) átrendezésével adódó (*10*) azt mutatja, hogy *m* értéke kiszámítható tetszőleges *n* érték esetén, csak nem garantálható, hogy ez az m egész szám legyen. Éppen ezt fogja az algoritmusunk ellenőrizni, hogy a pontos egyenlőséget biztosító számláló (*s*) egész szám-e, mert csak ekkor tekinthető megfelelőnek ($m = s$)

$$\mathrm{x} = \frac{s}{n} \cdot y \implies s = \frac{n \cdot x}{y} \qquad (10)$$

ahol:
$x =$ a felismerendő szám
$s =$ nem feltétlenül egész szám
$n =$ pozitív egész szám $(1 \leq n)$
$y =$ általánosított érték

Az alábbi kereső függvényünk azért kapta a `nevezo` nevet, mert azt a nevező értéket keresi és adja vissza eredményként, mely a kívánt megjelenítést biztosítja. Ha nem talál megfelelő nevezőt, akkor 0 lesz az eredmény.

```
Public Function nevezo(x, y)
  Dim n As Integer
  n = 1
  s = Abs(n * x / y)
  d = Abs(WorksheetFunction.Round(s, 0) - s)
  While n < 100 And d > 0.000001
    n = n + 1
    s = Abs(n * x / y)
    d = Abs(WorksheetFunction.Round(s, 0) - s)
  Wend
  If n < 100 Then
    nevezo = n
  Else
    nevezo = 0
  End If
End Function
```

A függvény úgy ellenőrzi, hogy a (*9*) egyenlőséget biztosító *s* érték egész-e, hogy megvizsgálja az egészre kerekített értékétől való eltérésének abszolútértékét, mint differenciát (`d`). Persze ennek 0-nak kellene lennie, ha s egész, de a számítógépes számábrázolás pontatlansága miatt 0-nak tekinthető egy nem negatív érték, ha nem nagyobb 0,000001-nél. Az egészre kerekítés a `Round` függvénnyel történik. A függvényünk az 1 értéktől kezdve egyesével növekvőleg keres alkalmas nevezőt, legfeljebb 100-ig. Így ha talál, akkor a legkisebbet fogja megtalálni.

A következő függvény azért kapta a `szamlalo` nevet, mert a (*9*) alakú előállítás számlálóját (*m*) határozza meg abban az esetben, ha ez az előállítás létezik, vagyis találtunk alkalmas nevezőt. A számláló 0 értékű lesz, ha nincs ilyen nevező.

```
Public Function szamlalo(x, y)
  Dim n, m As Integer
  n = nevezo(x, y)
  If n = 0 Then
    szamlalo = 0
  Else
    m = n * x / y
    szamlalo = m
  End If
End Function
```

Még azokra a függvényekre van szükségünk, mellyel megtudhatjuk, hogy a felismerendő szám (x) előállítható-e a Pi vagy a gyök jelek egyikével, azaz a (*3*) vagy (*4*) formában. Ezek a `pi_alap` és `gyok_alap` nevű függvények, melyek 0 értéket eredményeznek, ha nem lehetséges a kívánt alakú előállítás. Az előállíthatóságot a `nevezo` függvény értékéből tudhatjuk meg az $y = \pi$ vagy az $y = \sqrt{a}$

alkalmasságából. Előállíthatóság esetén a függvények értéke az előállíthatóságot biztosító érték lesz, vagyis a Pi érték a `pi_alap` függvénynél, illetve az a érték a `gyok_alap` függvénynél. Viszont ez utóbbi szintén egy kereső algoritmus a megfelelő a értékre, mely 1-től kezdve egyesével vizsgálandó. Így találat estén a legkisebb alkalmas értéket kapjuk még akkor is, ha több is alkalmas lett volna.

```
Public Function pi_alap(x)
  Dim n As Integer
  y = WorksheetFunction.Pi
  n = nevezo(x, y)
  If n = 0 Then
    pi_alap = 0
  Else
    pi_alap = y
  End If
End Function

Public Function gyok_alap(x)
  Dim a, n As Integer
  a = 1
  n = nevezo(x, a)
  While n = 0 And a < 100
    a = a + 1
    n = nevezo(x, a ^ (1 / 2))
  Wend
  If a < 100 Then
    gyok_alap = a
  Else
    gyok_alap = 0
  End If
End Function
```

Az eddig elkészített függvényeket használjuk fel a végső, a felismerendő szám (*3*) vagy (*4*) alakú megjelenítését eredményező, `txt_szam` nevű függvényünknél. Ennek eredménye szöveges adattípusúként a megfelelő alak, ha az létezik, különben a szám 4 tizedesjeggyel.

```
Public Function txt_szam(x)
  Dim s, n As Integer
  If x = 0 Then
    txt_szam = "0"
    Exit Function
  End If
  st = ""
  p = WorksheetFunction.Pi
  If pi_alap(x) > 0 Then
    s = szamlalo(x, p)
```

```
    n = nevezo(x, p)
    jel = ChrW(960)                'Pi karakter
  ElseIf gyok_alap(x) > 0 Then
    a = gyok_alap(x)
    If a = 1 Then
      y = 1
      jel = ""
    Else
      y = a ^ (1 / 2)
      jel = ChrW(8730) & a     'gyök jel
    End If
    s = szamlalo(x, y)
    n = nevezo(x, y)
  Else
    txt_szam = "" & WorksheetFunction.Round(x, 4)
    Exit Function
  End If
  If s = -1 Then
    If jel = "" Then
      st = "-1"
    Else
      st = "-" & jel
    End If
  ElseIf s = 1 Then
    If jel = "" Then
      st = "1"
    Else
      st = jel
    End If
  Else
    st = s & jel
  End If
  If n > 1 Then
    st = st & " / " & n
  End If
  txt_szam = st
End Function
```

A megjelenítés alakja több feltételtől függ:

- Előállítható-e a szám, s ha igen, akkor melyik alakban.
- Ha a számláló értéke 1, akkor csak olyan esetben szükséges a megjelenítése, ha a felismert érték racionális (például 1/4). Irracionális esetben elegendő az 1 érték nélkül a Pi vagy a gyökérték megjelenítése a számlálóban (például π/4 vagy √3/2).
- A nevezőt nem kell megjeleníteni, ha 1 az értéke, de ekkor a számláló szükséges olyan esetben, ha 1 számlálójú és racionális értékű.

- Negatív szám esetén kell az előjel, különben nem.

Az *1. táblázatban* néhány szám esetén látható a függvényünkkel kapott megjelenítésük.

1. táblázat: **Néhány felismert szám és megjelenítésük**

A szám értéke	A szám megjelenítése
1,13137085	4√2 / 5
12,12435565	7√3
4,188790205	4π / 3
4,294117647	73 / 17

Forrás: a szerző saját szerkesztése.

4. Következtetések

A Microsoft Excel táblázatkezelő programhoz olyan munkalapfüggvényt alakítottunk ki, mely képes a legfontosabb típusú irracionális számok esetén is a pontatlan és nehezen értelmezhető tizedes tört alak helyett olyan megjelenítésükre, mely jobban áttekinthető, rövidebb és a pontos értéket jelzi. A felismerhető számok köre igényeinknek megfelelően bővíthető például a harmadik és negyedik gyök segítségével megjeleníthető irracionális számokkal. A kifejlesztett függvényt bármely Excel táblázatunkban felhasználhatjuk az eredmények könnyebb értelmezéséhez, ahol ez szükséges lehet.

Irodalomjegyzék

Hampel Gy. (2017): Excel VBA alkalmazása egy biometriai esettanulmány példáján bemutatva, *Jelenkori társadalmi és gazdasági folyamatok*, 12 (4): 35–40.

Hampel Gy. (2018): Egymintás t-próba programozható kialakítása Excel VBA környezetben, Jelenkori társadalmi és gazdasági folyamatok, 13 (3-4): 169–175.

Kovalcsik G. (2005): *Az Excel programozása*. Computerbooks, Budapest.

Matteson B. L. (1995): *Microsoft Excel Visual Basic Programmer's Guide*. MicrosoftPress, Washington.

Obádovics J. Gyula (2012): *Matematika. Középiskolai tanulók, főiskolai és egyetemi hallgatók, valamint műszaki és gazdasági szakemberek számára, gyakorlati alkalmazásokkal.* Tizenkilencedik, bővített kiadás. Scolar Kiadó, Budapest.

Walkenbach, J. (2013): *Excel VBA Programming for Dummies*. 3rd edition. John Wiley & Sons Inc., New Jersey.