3 Feladatok

3.1 Programozási mintagyakorlatok alapvető algoritmusokkal

3.1.1 Tömb elemeinek számolása

3.1.1.1 „n” elemű numerikus tömb elemeinek összege

Program

```
Sub osszeg()
    Dim sngTomb(20) As Single
    Dim intN As Integer
    Dim intSum As Single
    Dim intI As Integer

    intN = 7
    intSum = 0

    For intI = 1 To intN
        sngTomb(intI) = Cells(1, intI)
        intSum = intSum + sngTomb(intI)
    Next intI

    Cells(1, intN + 1) = intSum
    Cells(2, 1) = „n tömb elemeinek összege”
End Sub
```

Megjegyzés

Tömb elemeinek összege

	A	B	C	D	E	F	G	H
1	1	2	3	4	5	6	7	28
2	n tömb elemeinek Összege							

tömb beolvasása cellákból
egydimenziós tömb elemeinek összegzése

3.1.1.2 „n × m” elemű tömb elemeinek összege (összes, - oszlop, - sor, - átló)

Program

```
Sub osszes_osszeg()
    Dim sngTomb(20, 20) As Single
    Dim intN%, intM%, intI%, intJ%
    Dim intSum!
    intN = 5: intM = 7
    intSum = 0
    For intI = 1 To intN
        For intJ = 1 To intM
            sngTomb(intI, intJ) = Cells(intI, intJ)
            intSum = intSum + sngTomb(intI, intJ)
        Next intJ
    Next intI
    Cells(intN + 2, 1) = „n tömb elemeinek Összege”
    Cells(intN + 1, intM + 1) = intSum
    End Sub
Sub oszlop_osszeg()
    Dim sngTomb(20, 20) As Single
    Dim intN%, intM%, intI%, intJ%
    Dim intSum_o(20) As Single
    intN = 5: intM = 7
    For intI = 1 To intM
        intSum_o(intI) = 0
```

Megjegyzés

Tömb elemeinek összege

a tömb
a tö
számol

	A	B	C	D	E	F	G	H
1	1	2	3	4	5	6	7	
2	1	2	3	4	5	6	7	
3	1	2	3	4	5	6	7	
4	1	2	3	4	5	6	7	
5	1	2	3	4	5	6	7	
6								140
7	n tömb elemeinek Összege							

tömboszlopok elemeinek összege

	A	B	C	D	E	F	G
1	1	2	3	4	5	6	7
2	1	2	3	4	5	6	7
3	1	2	3	4	5	6	7
4	1	2	3	4	5	6	7
5	1	2	3	4	5	6	7
6	5	10	15	20	25	30	35
7	a oszlopok elemeinek Összege						

```
        For intJ = 1 To intN                                    a tömb beolvasása cellákból
            sngTomb(intJ, intI) = Cells(intJ, intI)
            intSum_o(intI)      =      intSum_o(intI)      +
sngTomb(intJ, intI)
        Next intJ
        Cells(intN + 1, intI) = intSum_o(intI)
    Next intI
    Cells(intN + 2, 1) = „az oszlopok elemeinek összege”
End Sub
Sub sor_osszeg()
    Dim sngTomb(20, 20) As Single                               a tömbsorok elemeinek összege
    Dim intSum_s(20) As Integer
    Dim intN%, intM%, intI%, intJ%
    intN = 5
    intM = 7
    For intI = 1 To intN
        For intJ = 1 To intM                                    a tömb beolvasása cellákból
            sngTomb(intI, intJ) = Cells(intI, intJ)             a tömb sorösszegeinek számolása
            intSum_s(intI) = intSum_s(intI) + sngTomb(intI,
intJ)
        Next intJ
        Cells(intI, intM + 1) = intSum_s(intI)
    Next intI
    Cells(intN + 2, 1) = „a sorok elemeinek összege”
End Sub
Sub atlo_osszeg()
    Dim intSum_atlo as Integer
    Dim intN%, intM%, intI%, intJ%                              tömb
    intN = 7: intSum_atlo = 0                                   elemein
    For intI = 1 To intN
        intSum_atlo = intSum_atlo + Cells (intI, IntI)          a tömb a
    Next intI
    Cells(intN +1, intN+ 1) = intSum_atlo
    Cells(intN  +  2,1)  =  „Tömb  bal  átlója  elemeinek
összege”
End Sub
```

	A	B	C	D	E	F	G	H
1	1	2	3	4	5	6	7	28
2	1	2	3	4	5	6	7	28
3	1	2	3	4	5	6	7	28
4	1	2	3	4	5	6	7	28
5	1	2	3	4	5	6	7	28
6								
7	a sorok elemeinek Összege							

	A	B	C	D	E	F	G	H
1	1	2	3	4	5	6	7	
2	1	2	3	4	5	6	7	
3	1	2	3	4	5	6	7	
4	1	2	3	4	5	6	7	
5	1	2	3	4	5	6	7	
6	1	2	3	4	5	6	7	
7	1	2	3	4	5	6	7	
8								28
9	Tömb bal átlója elemeinek Összege							

3.1.1.3 Tömbváltozó elemei közötti Min és Max értékek keresése

Tömbváltozóban megkereshetőek e legnagyobb (Max) és legkisebb (Min) értékek, illetve rögzíthető a tömbben elfoglalt pozíciójuk is. A mintapélda rutin, egy dimenziós tömb cellából történő feltöltésével kezdődik. Ezt követően a Max és Min váltózónak „kötelezően” kezdő értéket kell adni, amely értékadásnál legcélszerűbb a vizsgált „tömb” első elemének értékét megadni, mert ez az elem biztosan a tömb elemei között szerepel, így a többi elem hozzá hasonlítható. Ezután egy For … Next ciklusban, ismételten If … Then parancsban vizsgálni lehet a „tömb aktuális elemének” > vagy < értékeit, s találat esetén tárolni kell az értéket és a találat helyét.

Program | Megjegyzés

```
Sub min_max()
    Dim intA() As Integer                                       dinamikustömb deklarációja
    Dim intMax%, intMaxh%
    Dim intMin%, intMinh%
    Dim intT%, intI%                                            célcellák (2. sor) előtörlése
    Rows(2).ClearContents
```

```
    intT = 0
    Do                                                  intA() tömb méretének növelése
       intT = intT + 1                                  intA() tömb feltöltése
       ReDim Preserve intA(intT)
       intA(intT) = Cells(1, intT)
    Loop While IsEmpty(Cells(1,intT+1)) = False         intMax és intMin kezdőérték megadása
    Cells(2, intT) = intT                               az integer adattípus értékkészlete alapján
    Cells(3, intT) = „Szám”                             intMax és intMin keresése
    intMax = intA(1): intMaxh = -32768
    intMin = intA(1): intMinh = 32767
    For intI = 1 To intT
       If intA(intI) < intMin Then
          intMin = intA(intI): intMinh = intI
       ElseIf intA(intI) > intMax Then
          intMax = intA(intI): intMaxh = intI
       End If
    Next intI
    Cells(2, intMaxh) = intMax
    Cells(3, intMaxh) = „Max”
    Cells(2, intMinh) = intMin
    Cells(3, intMinh) = „Min”
End Sub
```

	A	B	C	D	E	F	G	H	I	J	K
1	3	7	12	25	2	11	1	8	9	2	10
2				25			1				11
3				Max			Min				Szám

3.1.2 Sorfüggvények és Faktoriális számolás rutin eljárások

3.1.2.1 Sorfüggvény példa

Adott az $f(k)=\frac{3}{k^2+5k+6}$ függvény. Megírandó a „Sorfuggveny” VBA program, amely a $Sum=\sum_{k=1}^{m} f(k)=f(1)+f(2)+...+f(m)$ összeget számítja ki és az összeget a felhasználó által meghatározott „*p*” értéktől ki is írja az Excel cellákba.

Program

```
Sub Sorfuggveny()
Dim intM%, intK%, intP%
Dim sngFk!, sngSum!
intM = InputBox(„m = ?”, „Adatbekeres”, 5)
intP = InputBox(„p = ?”, „Adatbekeres”, 3)
sngSum = 0
Cells(1, 1) = „k”: Cells(1, 2) = „fk”: Cells(1, 3) =
„sum”
For intK = 1 To intM
   Cells(intK + 3, 1) = intK
   sngFk = 3 / (intK ^ 2 + 5 * intK + 6)
   Cells(intK + 3, 2) = Round(sngFk, 3)
   sngSum = sngSum + sngFk

   If intK >= intP Then
      Cells(intK + 3, 3) = Round(sngSum, 3)
   End If

Next intK

End Sub
```

Megjegyzés

Beolvasások InputBox paranccsal

	A	B	C
1	k	fk	sum
2			
3			
4	1	0,250	
5	2	0,150	
6	3	0,100	0,500
7	4	0,071	0,571
8	5	0,054	0,625

ekít és cellába ír.

sának feltételét vizsgálja.

3.1.2.2 Faktoriális számítása

A Faktorialis programot beolvassa x és n értékeket, majd kiszámolja a

$$1, x, \frac{x^2}{2!}, \frac{x^3}{3!}, \ldots, \frac{x^n}{n!}$$

faktoriális sorfüggvény tagjainak értékét és a tagok összegét. Megfigyelhető, hogy a tagok egymásból $\frac{x}{i}$ értékkel való szorzással állíthatóak elő, ahol az $1 < i < n$ egész számok halmaza.

Program

```
Sub faktorialis()
Dim intX%, intN%, sngTag!
Dim sngSum!, intI%
intN = InputBox(„N= ?”, „Beolvasás”, 11)
intX = InputBox(„X= ?”, „Beolvasás”, 3)
Cells(1, 1) = „N”: Cells(1, 2) = „X”
Cells(1, 3) = „Tag”
sngTag = 1: Cells(2, 3) = sngTag
sngSum = 0
For intI = 1 To intN
   sngTag = sngTag * (intX / intI)
   Cells(intI + 2, 1) = intI
   Cells(intI + 2, 3) = sngTag
   sngSum = sngSum + sngTag
Next intI
   Cells(intN + 3, 2) = „Sum”
   Cells(intN + 3, 3) = sngSum
End Sub
```

Megjegyzés

tagok ö

	A	B	C
1	N	X	Tag
2			1
3	1		3
4	2		4,5
5	3		4,5
6	4		3,375
7	5		2,025
8	6		1,0125
9	7		0,433929
10	8		0,162723
11	9		0,054241
12	10		0,016272
13	11		0,004438
14		Sum	19,0841

3.1.3 Műveletek vektorokkal és tömbökkel

3.1.3.1 Vektorok összege, szorzata, hossza

Összeg: $A + B = (A(1) + B(1)), (A(2) + B(2)), \ldots (A(N) + B(N))$

Szorzat: $A * B = (A(1) * B(1)) + (A(2) * B(2)) + \ldots (A(N) * B(N))$

Hossz: $A\ hossz = Sqr((A(1) * B(1)) + (A(2) * B(2)) + \ldots (A(N) * B(N)))$

Program

```
Sub vektor()
Dim intA%(3), intB%(3)
Dim intC%(3), intE%, intI%
For intI = 1 To 3
   intA(intI) = Cells(3, intI + 1)
   intB(intI) = Cells(5, intI + 1)
   intC(intI) = intA(intI) + intB(intI)
   Cells(7, intI + 1) = intC(intI)
Next intI
intE = 0
```

Megjegyzés

A vektor beolvasása
B vektor beolvasása
A + B összeadás
A + B kiírása

A * B számolás

	A	B	C	D
1	Vektor			
2				
3	A	4	2	9
4				
5	B	2	-1	6
6				
7	A + B	6	1	15
8				
9	A * B	60		
10				
11	A hossz	10,04988		
12				
13	B hossz	6,403124		

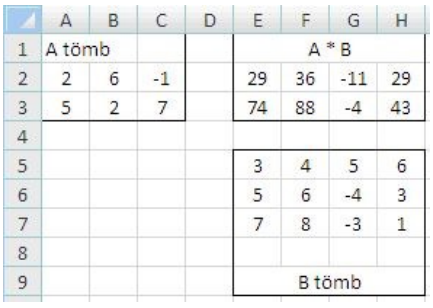
```
For intI = 1 To 3
   intE = intE + intA(intI) * intB(intI)
Next intI
Cells(9, 2) = intE                    A hossz négyzetének számolása
intE = 0
For intI = 1 To 3
   intE = intE + intA(intI) ^ 2       B hossz négyzetének számolása
Next intI
Cells(11, 2) = Sqr(intE)
intE = 0
For intI = 1 To 3
   intE = intE + intB(intI) ^ 2
Next intI
Cells(13, 2) = Sqr(intE)
End Sub
```

3.1.3.2 Tömbök szorzása

A tömbök szorzásának feltétele, hogy egyik dimenziójuk nagysága azonos kell, hogy legyen. A mintapélda egy A(2 x 3) tömb szorzása B(3 x 4) tömbbel, melynek eredménye C(2 x 4) tömb. A tömbök elhelyezése az Excel lapon csak a „vizuális látvány” és az érthetőség segítését szolgálja.

Program — Megjegyzés

```
Sub tömb_szorzas()
Dim intA%(2, 3), intB%(4, 3), intC%(3, 4)
Dim intI%, intK%, intJ%
For intI = 1 To 2
   For intK = 1 To 3                  „A” tömb feltöltése
      intA(intI, intK) = Cells(intI + 1, intK)
   Next intK
Next intI                             „B” tömb feltöltése
For intI = 1 To 4
   For intK = 1 To 3                  „C” tömb helyének törlése
      intB(intI, intK) = Cells(intK + 4, intI +     3x  For … Next számoló
4)
   Next intK
Next intI
Range(„E2:F3”).ClearContents
For intI = 1 To 2
   For intJ = 1 To 4
      s = 0
      For intK = 1 To 3
         s=  s+(intA(intI,  intK)  *  intB(intJ,
intK))
      Next intK
      intC(intI, intJ) = s
      Cells(intI + 1, intJ + 4) = s
   Next intJ
Next intI
End Sub
```



	A	B	C	D	E	F	G	H
1	A tömb				A * B			
2	2	6	-1		29	36	-11	29
3	5	2	7		74	88	-4	43
4								
5					3	4	5	6
6					5	6	-4	3
7					7	8	-3	1
8								
9					B tömb			

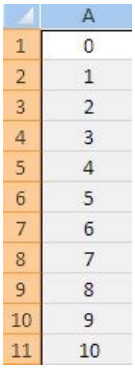
3.1.4 Kisebb – nagyobb rendezés (Buborékrendezés)

Program — Megjegyzés

```
Sub novekvo_rendezes()                A program futtatása előtt véletlenszerűen
Dim intK%, intI%                      ki kell tölteni az A1:A10 cellákat
```

```
Dim intA%, intB%, intC%
For intK = 1 To 10
   For intI = 1 To 10 - (intK - 1)
      intA = Cells(intI, 1)
      intB = Cells(intI + 1, 1)
      If intA > intB Then
         intC = intA
         Cells(intI, 1) = intB
         Cells(intI + 1, 1) = intC
      End If
   Next intI
Next intK
End Sub
```

az 1-10 számokkal.



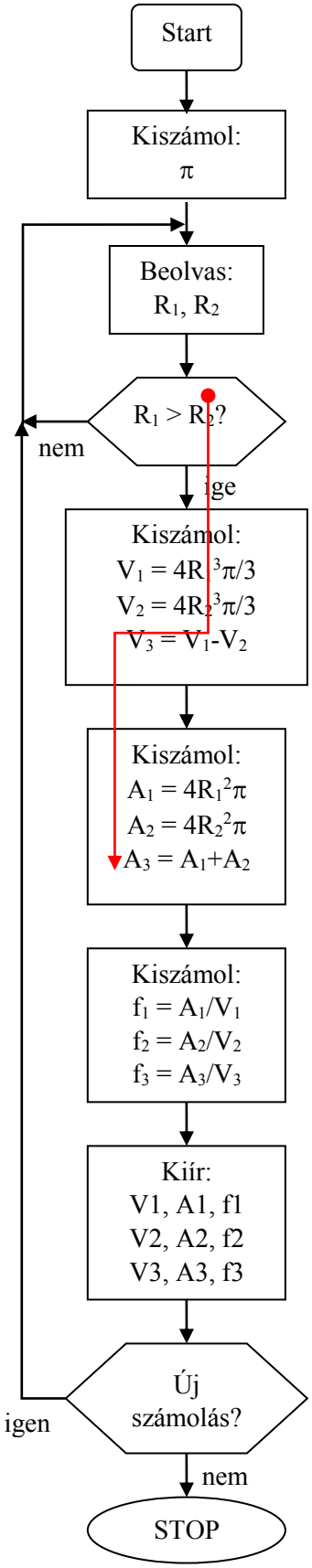
	A
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	10

3.2 Kidolgozott gyakorló feladatok

3.2.1 Program írása blokkdiagram alapján

3.2.1.1 Térfogatszámolás feladat

Írja meg a „Terfogatszamolas” nevű VBA programot a mellékelt blokkdiagram alapján az If … Then – GoTo alkalmazásával! Az adatbevitelhez az InputBox, üzenetkiíráshoz a MsgBox parancsot használja! Értelmezze a program működését! Indokolja meg, miért szükséges az R1 > R2 kitétel! A program megírása során az alábbi változókat alkalmazza: sngR1, sngR2, sngV1, sngV2, sngV3, sngA1, sngA2, sngA3, sngf1, sngf2, sngf3, sngPi (mind single) és strDontes (string). A ciklusok, tömbök, függvények és a szubrutinok alkalmazásának gyakorlása során térjen vissza a feladatra és egyszerűsítse a programot!



Megoldás:

Először a programnak kell nevet adni, majd ezt követi a változók deklarálása.

```
Sub Terfogatszamolas()
    Dim sngR1 As Single, sngR2!
    Dim sngV1 As Single, sngV2!, sngV3!
    Dim sngA1 As Single, sngA2!, sngA3!
    Dim sngf1 As Single, sngf2!, sngf3!
    Dim strDontes As String
    Dim sngPi As Single
```

A következő lépésben π értékét kell kiszámolni a matematikai függvényeknél leírtak alapján.

```
    sngPi = 4 * Atn(1)
```

A blokkdiagramon látszik, hogy mielőtt R_1 és R_2 értékét beolvasná a program, meg kell oldani a program végéről a visszavezetést, melyre az „ujra” címke szolgál:

```
ujra:
    sngR1 = InputBox(„R1 = ?”, „Adatbekeres”, 2)
    sngR2 = InputBox(„R2 = ? (kisebb, mint R1!)”, „Adatbekeres”, 1)
```

Ezek után le kell ellenőrizni, hogy $R_1 > R_2$, amennyiben a feltétel nem teljesül, R_1 és R_2 értékét újra be kell olvasni, amihez szintén az ujra címkét kell felhasználni:

```
    If sngR2 >= sngR1 Then GoTo ujra
```

Ezt követi a térfogatok (V) és a felületek (A) és a fajlagos felületek (f) kiszámolása:

```
sngV1 = 4 * sngR1 ^ 3 * sngPi / 3
sngV2 = 4 * sngR2 ^ 3 * sngPi / 3
sngV3 = sngV1 - sngV2
sngA1 = 4 * sngR1 ^ 2 * sngPi
sngA2 = 4 * sngR2 ^ 2 * sngPi
```

```
sngA3 = sngA1 + sngA2
sngf1 = sngA1 / sngV1
sngf2 = sngA2 / sngV2
sngf3 = sngA3 / sngV3
```

Az „1” és „2” indexű változók egy-egy gömb térfogatára, felületére és fajlagos felületére vonatkoznak, míg a „3” indexű változók a két gömb tulajdonságainak különbségére, azaz a nagyobbik gömb „héjára” (például narancshéj) vonatkoznak. A térfogat csak pozitív szám lehet, ezért szükséges, hogy $R_1 > R_2$ legyen. Ezután a MsgBox paranccsal ki kell írni a számolt értékeket. A szöveg és szám típusú változók összefűzésére a & karakter szolgál. Nem szabad elfelejteni, hogy a kiírandó szöveget idézőjelek közé kell tenni!

```
MsgBox „V1 = „ & sngV1 & „„ A1 = „ & sngA1 & „„ f1 = „ & sngf1, , „1. adatainak kiirasa”
MsgBox „V2 = „ & sngV2 & „„ A2 = „ & sngA2 & „„ f2 = „ & sngf2, , „2. adatainak kiirasa”
MsgBox „V3 = „ & sngV3 & „„ A3 = „ & sngA3 & „„ f3 = „ & sngf3, , „3. adatainak kiirasa”
```

Végezetül meg kell kérdezni, hogy a felhasználó kíván új számolást végezni vagy nem. Igen válasz esetén vissza kell térni az „ujra” címkéhez, míg nem válasz esetén a program véget ér.

```
    strDontes = InputBox(„Uj szamolas? (i = igen, bármi más = nem)”, „Adatbekeres”, „i”)
    If strDontes = „i” Then GoTo ujra
    MsgBox „Program vege”
End Sub
```

3.2.2 Program írása blokkdiagram nélkül

3.2.2.1 Öröknaptár feladat

Írja meg az „Oroknaptar” nevű programot, amely megadja, hogy egy adott dátum a hét melyik napjára esett/fog esni. Referenciadátum: 2009. 08. 31., hétfő. A megoldás során használja a maradékosztásra szolgáló „mod” parancsot, a „Select Case” utasítást, valamint az alábbi változókat: datReferencia, datKerdesesNap (mindkettő date), sngKulonbseg (single, a két dátum különbsége), intMaradek (integer, maradékosztás eredménye). Ne feledkezzen el arról, hogy sngKulonbseg és intMaradek értéke negatív is lehet! Miért nem használhatunk az sngKulonbseg változó helyett integer típusú intKulonbseg változót? Az intMaradek változó helyett alkalmazhatnánk byte típusú bytMaradek változót? A tömbökről tanultak után térjen vissza a feladatra, írja át a megoldást!

Megoldás:

Először a változókat kell deklarálni:

```
Sub Oroknaptar()
Dim datReferencia As Date, datKerdesesNap As Date
Dim intMaradek As Integer
Dim sngKulonbseg As Single
```

Ezt követi a referenciadátum megadása, illetve a kérdéses nap dátumának bekérése a felhasználótól:

```
datReferencia = #8/31/2009#
datKerdesesNap = InputBox(„Kerem, adja meg a kerdeses nap datumat!”, „Adatbekeres”, #1/1/2009#)
```

A dátumokat a Visual Basic számokként kezeli, így különbségük is képezhető. A különbség lehet pozitív, illetve negatív szám is, mert a kérdéses nap dátuma lehet a referencianapnál későbbi, illetve korábbi dátum is. A különbséget el kell osztani héttel és meg kell határozni a maradékot, mely egyértelműen megadja, hogy a kérdéses dátum milyen napra esett.

```
sngKulonbseg = datKerdesesNap - datReferencia
intMaradek = sngKulonbseg Mod 7
Select Case intMaradek
    Case 0
        MsgBox „A kerdeses nap hetfo volt/lesz.”
    Case -6, 1
        MsgBox „A kerdeses nap kedd volt/lesz.”
    Case -5, 2
        MsgBox „A kerdeses nap szerda volt/lesz.”
    Case -4, 3
        MsgBox „A kerdeses nap csutortok volt/lesz.”
    Case -3, 4
        MsgBox „A kerdeses nap pentek volt/lesz.”
    Case -2, 5
        MsgBox „A kerdeses nap szombat volt/lesz.”
    Case -1, 6
        MsgBox „A kerdeses nap vasarnap volt/lesz.”
End Select
End Sub
```

3.2.3 Gyakorlatok For - Next és Do - Loop ciklusok alkalmazására

3.2.3.1 Lottó feladat

Írjon VBA programot a For-Next és a Do-Loop (elől és hátul tesztelő) ciklus alkalmazásával, amelynek segítségével kiszámítható a nyerési esély a Magyarországon forgalmazott lottók (5/90, 6/45) esetében. Az adatok beolvasásához az InputBox (a lottószelvényen lévő számok és a kihúzandó számok mennyisége, bytN és bytK, mindkettő byte típusú változó), az adatok kiírásához a MsgBox parancsot használja. A nyerési esély a $\frac{n!}{(n-k)!k!}$ képlet segítségével számítható ki. A gyorsabb számolás érdekében a fenti képlet alábbi egyszerűsített változatát használja: $\frac{n(n-1)...(n-k+1)}{k!} = \prod_j \frac{n-j}{1+j}$; ahol $j = 0,1,...,k-1$. A programok írásakor használjon modulszintű változókat.

Megoldás:

A modulszintű változókat célszerű az „Option” parancsok alatt, de az első program nyitó „Sub” sora felett deklarálni:

```
Option Explicit
Dim bytJ As Byte, bytK As Byte, bytN As Byte
Dim sngNyeresiEsely As Single
Sub Lotto_For()
```

A sngNyeresiEsely változó tartalmazza majd az összes lehetséges húzásvariáció számát, míg a bytJ változó a For - Next ciklus ciklusváltozója. A deklarációt követi a kihúzható lottószámok, valamint a húzások számának megadása:

```
bytN = InputBox(„Kihuzhato lottoszamok szama (max. 255)”, „Adatbekeres”, 90)
bytK = InputBox(„Huzasok szama (max. 255)”, „Adatbekeres”, 5)
```

A következő lépésben a sngNyeresiEsely változónak kiindulási értéket kell adni, csak ezután következhet a For - Next ciklus írása. A képlet szerint a For - Next ciklusban a törteket szorozni kell egymással, így sngNyeresiEsely változó kiindulási értéke 1:

```
sngNyeresiEsely = 1
For bytJ = 0 To bytK - 1
   sngNyeresiEsely = sngNyeresiEsely * (bytN - bytJ) / (1 + bytJ)
Next bytJ
```

Végezetül a kapott eredményt kell kiírni:

```
MsgBox „Az adott lottonal a nyeresi esely = 1 : „ & sngNyeresiEsely & „.”, , „Eredmeny”
End Sub
```

Természetesen a feladat Do Loop ciklussal is megoldható, ebben az esetben a

```
For bytJ = 0 To bytK - 1
   sngNyeresiEsely = sngNyeresiEsely * (bytN - bytJ) / (1 + bytJ)
Next bytJ
```

programrészlet helyett elől tesztelő Do - Loop ciklust alkalmazva a

```
bytJ = 0
Do While bytJ <= bytK - 1
   sngNyeresiEsely = sngNyeresiEsely * (bytN - bytJ) / (1 + bytJ)
   bytJ = bytJ + 1
Loop
```

programrészletet, míg hátul tesztelő Do - Loop ciklust alkalmazva a

```
bytJ = 0
Do
   sngNyeresiEsely = sngNyeresiEsely * (bytN - bytJ) / (1 + bytJ)
   bytJ = bytJ + 1
Loop While bytJ <= bytK - 1
```

programrészletet kell használni. Látható, hogy a For - Next ciklushoz képest a Do - Loop ciklus használatakor a ciklus előtt bytJ változónak kiindulási értéket kell adni, illetve a ciklus futása során bytJ értékét meg kell növelni.

3.2.3.2 Püthagorasz-féle számhármasok meghatározása

Írjon VBA programot Do-Loop (előltesztelő)és a For-Next ciklus alkalmazásával. amely meghatározza az összes olyan Pithagorasz-féle számhármast, amelyben a három egész szám egyike sem nagyobb, mint 100. A három szám (intA, intB és intC (integer típusú változók)) értékeit külön oszlopokban, a 2. sortól kezdve írassa ki egymás alatti cellákba (Az első sor fejlécként az „A”, „B” és „C” betűket tartalmazza.). A programok írásakor használjon modulszintű változókat.

Megoldás:

A program írása előtt értelmezni kell a feladat szövegét. A derékszögű háromszög leghosszabb oldala az átfogó (intC), így a „három egész szám egyike sem nagyobb, mint 100” kitétel gyakorlatilag azt jelenti, hogy intC ≤ 100. A két befogó (intA, intB) közül az egyik befogó (intB) hosszának nagyobbnak kell lennie, mert intA = intB esetén az $intA^2 + intB^2 = intC^2$ összefüggés értelmében $intC^2 = 2 \times intB^2$, amiből egyértelmű, hogy intC és intB egyszerre nem lehet egész szám. A fentebb leírtak alapján intA < intB < intC. A feladat legegyszerűbben három, egymásba ágyazott ciklus segítségével oldható meg, ahol intC értéke a 3 - 100, intB értéke a 2 - intC-1, míg intA értéke a 1 - intB-1 tartományban változhat.

```
Sub Pithagorasz_For()
Dim intA As Integer, intB%, intC%, intI%
```

intI változóra intA, intB és intC egymás alatti sorokban történő kiírása miatt van szükség.

```
Cells(1, 1) = „A”
Cells(1, 2) = „B”
Cells(1, 3) = „C”
```

Az A1:A3 cellákba kerül a kiírandó adatok táblázatának fejléce. A következő lépésben intI változónak kell kiindulási értéket adni. Az első Pithagorasz-féle számhármast a B1:B3 cellákba (azaz a második sorba) kell kiírni, ezután következhet az egymásba ágyazott For-Next ciklusokmegnyitása:

```
intI = 2
For intC = 3 To 100
    For intB = 2 To intC - 1
        For intA = 1 To intB - 1
```

A következő lépésben meg kell vizsgálni, hogy teljesül-e az $intA^2 + intB^2 = intC^2$ feltétel. Amennyiben a feltétel teljesül, akkor a változók értékeit ki kell írni az Excel munkalapra, illetve intI értékét meg kell növelni, hogy a következő számhármas kiírása ne ugyanabba a sorba történjen. Ezt követi a ciklusok zárása:

```
            If intC ^ 2 = intA ^ 2 + intB ^ 2 Then
                Cells(intI, 1) = intA
                Cells(intI, 2) = intB
                Cells(intI, 3) = intC
                intI = intI + 1
            End If
        Next intA
    Next intB
Next intC
End Sub
```

Látható, hogy egymásba ágyazott ciklusok esetén először a legbelső ciklust kell bezárni, míg utolsónak a legkülső ciklus bezárása marad.

A programon lehet finomítani, hiszen a harmadik ciklusra tulajdonképpen nincs szükség, mert a Püthagorasz tétel alapján két oldal hosszának ismeretében a harmadik oldal hossza számítható. Ugyanakkor az így számított érték a legtöbbször nem egész szám, tehát a programozás során egy új, single típusú változót (sngA) is deklarálni kell:

```
Sub Pithagorasz_For_2()
    Dim intB%, intC%, intI%
    Dim sngA as Single
    Cells(1, 1) = „A”
    Cells(1, 2) = „B”
    Cells(1, 3) = „C”
    intI = 2
    For intC = 3 To 100
        For intB = 2 To intC - 1
```

Eddig a program megegyezik az előző programmal, de most kerül sor sngA értékének kiszámítására és annak vizsgálatára, hogy sngA egész szám vagy sem:

```
            sngA = Sqr(intC ^ 2 - intB ^ 2)
            If sngA = Int(sngA) And intB > Int(sngA) Then
                Cells(intI, 1) = sngA
                Cells(intI, 2) = intB
                Cells(intI, 3) = intC
                intI = intI + 1
            End If
        Next intB
    Next intC
    End Sub
```

Az Sqr függvény a gyökvonást végzi. Természetesen jó megoldás a

```
            sngA = (intC ^ 2 - intB ^ 2) ^ 0.5
```

parancs is. Az Int(sngA) kifejezés sngA értékének egész részét állítja elő, amely természetesen csak akkor lehet egyenlő sngA értékével, ha sngA egész szám. Az intB > Int(sngA) feltételre nem csak azért van szükség, mert korábban kikötöttük, hogy intB > intA, hanem azért is, hogy a program ugyanazt a megoldást ne kétszer találja meg (intA = 3, intB = 4, intC = 5; illetve intA = 4, intB = 3, intC = 5).

A feladat természetesen megoldható Do - Loop ciklus alkalmazásával is:

```
Sub Pithagorasz_Do()
    Dim intB%, intC%, intI%
    Dim sngA as Single
        Cells(1, 9) = „A”
        Cells(1, 10) = „B”
        Cells(1, 11) = „C”
    intC = 3
    intI = 2
    Do While intC <= 100
        intB = 2
        Do While intB < intC
            sngA = Sqr(intC ^ 2 - intB ^ 2)
            If sngA = Int(sngA) And intB > Int(sngA) Then
                Cells(intI, 9) = sngA
                Cells(intI, 10) = intB
                Cells(intI, 11) = intC
                intI = intI + 1
            End If
            intB = intB + 1
        Loop
        intC = intC + 1
    Loop
```

```
End Sub
```

Látható, hogy a Püthagorasz_Do program csak annyiban Püthagorasz_For_2 programtól, hogy előbbiben a Do - Loop ciklusok alkalmazása előtt intC és intB változóknak kiindulási értéket kell adni (3, illetve 2), valamint intC és intB változók értékét a ciklusuk vége előtt meg kell növelni.

3.2.3.3 Számkitaláló feladat

Írjon olyan VBA programot Do-Loop (hátul tesztelő) ciklus alkalmazásával, amelyben a számítógép kigondol egy 1 és 100 közé eső egész számot (intGondol, integer), majd felszólítja a felhasználót, hogy találja ki azt (MsgBox utasítás). A felhasználó tippjére (intTipp, integer; InputBox utasítás) közli, hogy az nagyobb, vagy kisebb az általa gondoltnál, illetve a helyes tipp esetén gratulál és felajánlja, hogy új játékot kezdjenek (strDontes, string). Kérdés: A feladat megoldható For-Next ciklus alkalmazásával?

Megoldás:

A program írása előtt végig kell gondolni, hány ciklus alkalmazására van szükség. Ciklust kell alkalmazni akkor, amikor

- a program megkérdezi, hogy a felhasználó kíván-e új játékot játszani,
- a program bekéri a felhasználó következő tippjét.

A két ciklus közül egyértelműen a tipp bekérése a belső ciklus, hiszen a tippkérés a játékon **belül** történik, míg az új játékra vonatkozó ciklus a külső ciklus, mert az új játékra vonatkozó kérdés a játékon **kívül**, két játék között történik.

A program írása során először deklarálni kell a használt változókat:

```
Sub Kitalalo()
    Dim intGondol As Integer, intTipp%
    Dim strDontes As String
```

Ezt követi a külső ciklus megnyitása és a véletlen szám generálása:

```
Do
    intGondol = Int(100 * Rnd(1)) + 1
    MsgBox „Kerem, talalja ki melyik szamra gondoltam!”
```

A Rnd függvény a 0-1 tartományban generál számot, amit a feladat „1 és 100 közé eső egész szám” kitétele miatt előbb be kell szorozni százzal, majd az így kapott szám egész részét (0, 1, 2, … 99) kell venni az Int függvénnyel, végül meg kell az értékét növelni eggyel. Most kerülhet sorra a belső ciklus megnyitása majd a tipp bekérése:

```
    Do
        intTipp = InputBox(„Melyik szamra tippel?”, „Adatbekeres”)
```

A beadott tippet össze kell hasonlítani a számítógép által generált számával, majd közölni kell a felhasználóval, hogy a két érték hogyan viszonyul egymáshoz:

```
        Select Case intTipp
```

```
        Case Is < intGondol
            MsgBox „Nagyobb számra gondoltam!”
        Case Is > intGondol
            MsgBox „Kisebb számra gondoltam!”
        Case intGondol
            MsgBox „Gratulálok! Kitalálta a számot!”
    End Select
```

A Select Case - End Select szerkezet helyett az if - else - End If szerkezet is használható, ebben az esetben a fenti programrészlet helyett az alábbi programrészletet kell megírni:

```
    If intTipp < intGondol Then
        MsgBox „Nagyobb szamra gondoltam!”
        ElseIf intTipp > intGondol Then
            MsgBox „Kisebb szamra gondoltam!”
        Else
            MsgBox „Gratulalok! Kitalalta a szamot!”
    End If
```

Látható, hogy ebben az esetben már nincs értelme az Else után újabb feltételt megadni (bár lehet), hiszen, ha intTipp se nem nagyobb, se nem kisebb, mint intGondol, akkor csak egyenlő lehet vele. Most kerülhet sor a belső ciklus bezárására:

```
Loop Until intGondol = intTipp
```

Természetesen a

```
Loop While intGondol <> intTipp
```

parancs is megfelel. Mivel a belső ciklus bezárásával a játék is véget ért, a felhasználót meg kell kérdezni, hogy kíván új játékot kezdeni vagy sem. Ezt követi a külső ciklus bezárása és a program vége:

```
strDontes = InputBox(„Van kedve új játékra? (i = igen, bármi más = nem)”, „Adatbekérés”, „i”)
Loop While strDontes = „i”
End Sub
```

Természetesen a külső ciklus bezárása a

```
Loop Until strDontes <> „i”
```

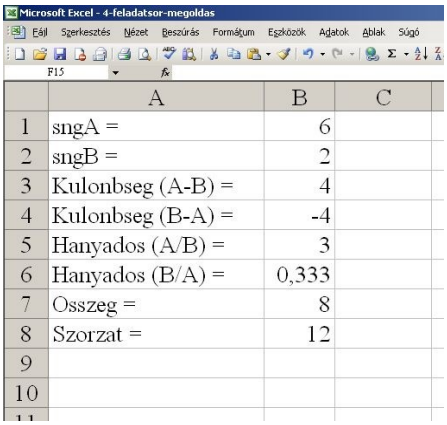
paranccsal is megtörténhet.

3.2.4 Gyakorlatok Function függvények és szubrutinok alkalmazására

3.2.4.1 Alapműveletek feladat

Írja meg az „Alapmuveletek” VBA programot, amely függvények (Function) felhasználásával kiszámítja két tetszőleges szám (sngA és sngB, pl: 6 és 2) összegét, különbségét, szorzatát és hányadosát. A kapott eredményeket a jobboldali ábrának megfelelően írassa ki.

Utána írja meg az „Alapmuveletek2” VBA programot, amely ugyanazt végzi, mint az „Alapmuveletek”, csak függvények helyett szubrutinokat használ. A programok írásakor használjon modulszintű változókat. (sngA, sngB, sngKulonbseg1, sngKulonbseg2, sngHanyados1 és sngHanyados2 single típusú változók, dblOsszeg és dblSzorzat double típusú változók).



	A	B	C
1	sngA =	6	
2	sngB =	2	
3	Kulonbseg (A-B) =	4	
4	Kulonbseg (B-A) =	-4	
5	Hanyados (A/B) =	3	
6	Hanyados (B/A) =	0,333	
7	Osszeg =	8	
8	Szorzat =	12	
9			
10			

Megoldás:

Első lépésben deklaráljuk a programok írása során használt változókat. Mivel mindkét program ugyanazokat a változókat használja, célszerű modulszintű változókat használni.

```
Option Explicit
Dim sngA As Single, sngB!, sngKulonbseg1!, sngKulonbseg2!
Dim sngHanyados1!, sngHanyados2!
Dim dblOsszeg As Double, dblSzorzat#
Sub Alapmuveletek()
```

A deklarációt sngA és sngB változók beolvasása, majd munkalapra történő írása követheti:

```
sngA = InputBox(„Kerem A erteket!”, „Adatbekeres”, 6)
sngB = InputBox(„Kerem B erteket!”, „Adatbekeres”, 2)
Cells(1, 1) = „sngA =„: Cells(1, 2) = sngA
Cells(2, 1) = „sngB =„: Cells(2, 2) = sngB
```

A megadott két szám értékének ismeretben meghívhatóak a különbség, összeg, hányados és szorzat értékeket számítható függvények. A függvényekben közös, hogy mindegyiknek át kell adni sngA és sngB értékét, míg a függvények az eredményt a nevükkel adják vissza:

```
sngKulonbseg1 = Kivon(sngA, sngB): sngKulonbseg2 = Kivon(sngB, sngA)
sngHanyados1 = Eloszt(sngA, sngB): sngHanyados2 = Eloszt(sngB, sngA)
dblOsszeg = Osszead(sngA, sngB)
dblSzorzat = Osszeszoroz(sngA, sngB)
```

A „Kivon” függvény második hívása helyett alkalmazható a

```
sngKulonbseg2 = -sngKulonbseg1
```

utasítás is. A kiszámolt értékek kiírhatóak az Excel munkalapra:

```
Cells(3, 1) = „Kulonbseg (A-B) =„: Cells(3, 2) = sngKulonbseg1
Cells(4, 1) = „Kulonbseg (B-A) =„: Cells(4, 2) = sngKulonbseg2
Cells(5, 1) = „Hanyados (A/B) =„
```

A hányados számolásánál gond lehet, amennyiben a nevezőben lévő szám értéke 0. Az sngA ≠ 0, sngB ≠ 0 feltételek vizsgálata történhet az „Osztas” függvény hívása előtt, vagy a függvényben. Ha a programban a függvényt többször hívják meg, célszerű lehet a vizsgálatot a függvényben elvégezni. Ez az oka, hogy a

```
sngHanyados2 = 1/ sngHanyados1
```

parancsok nem minden esetben jó megoldások. A hányadosok értékének munkalapra történő kiírásakor viszont jelezni kell a felhasználónak, ha a hányados nem számítható:

```
If sngB = 0 Then
    Cells(5, 2) = „Nem ertelmezheto”
Else
    Cells(5, 2) = sngHanyados1
End If
    Cells(6, 1) = „Hanyados (B/A) =„
If sngA = 0 Then
    Cells(6, 2) = „Nem ertelmezheto”
Else
    Cells(6, 2) = sngHanyados2
End If
```

Végezetül az Excel munkalapra a két szám összegének és szorzatának értékét kell kiírni:

```
    Cells(7, 1) = „Osszeg =„: Cells(7, 2) = dblOsszeg
    Cells(8, 1) = „Szorzat =„: Cells(8, 2) = dblSzorzat
End Sub
```

A főprogram megírása után kerülhet sor a függvények megírására. Noha mindegyik függvény sngAf és sngBf változókat használ, de ez nem okoz hibát, mert mindegyik változó hatóköre csak a saját függvényére terjed ki.

```
Function Kivon(sngAf!, sngBf!) As Single
    Kivon = sngAf - sngBf
End Function
Function Osszead(sngAf!, sngBf!) As Double
    Osszead = sngAf + sngBf
End Function
Function Osszeszoroz(sngAf!, sngBf!) As Double
    Osszeszoroz = sngAf * sngBf
End Function
Function Eloszt(sngAf!, sngBf!) As Single
    If sngBf = 0 Then
        MsgBox „Hiba! Oszto nem lehet 0!”
        Exit Function
    Else
        Eloszt = sngAf / sngBf
    End If
End Function
```

Látható, hogy a sngBf = 0 esetben az Eloszt függvény anélkül fejeződik be (Exit Function), hogy értéket a főprogram sngHanyados1 vagy sngHanyados2 változójának.

Az Alapműveletek2 program írásakor az Alapműveletek program jelentős része átvehető, eltérés csak abban van, hogy az Alapműveletek2 program függvények helyett szubrutinokat használ, melyeket a Call paranccsal kell meghívni:

```
Call Kivonasok(sngA, sngB, sngKulonbseg1, sngKulonbseg2)
Call Osztasok(sngA, sngB, sngHanyados1, sngHanyados2)
Call Osszeadas(sngA, sngB, dblOsszeg)
Call Osszeszorzas(sngA, sngB, dblSzorzat)
```

Végezetül a szubrutinokat kell megírni.

```
Sub Kivonasok(sngAf!, sngBf!, sngEredmeny1!, sngEredmeny2!)
    sngEredmeny1 = sngAf - sngBf
    sngEredmeny2 = sngBf - sngAf
End Sub
```

```
    Sub Osszeadas(sngAf!, sngBf!, dblEredmeny1!)
    dblEredmeny1 = sngAf + sngBf
End Sub
Sub Osszeszorzas(sngAf!, sngBf!, dblEredmeny1#)
    dblEredmeny1 = sngAf * sngBf
End Sub
Sub Osztasok(sngAf!, sngBf!, sngEredmeny1!, sngEredmeny2!)
    If sngBf = 0 Or sngAf = 0 Then MsgBox „Hiba! Oszto nem lehet 0!”: Exit Sub
    If sngBf <> 0 Then sngEredmeny1 = sngAf / sngBf
    If sngAf <> 0 Then sngEredmeny2 = sngBf / sngAf
End Sub
```

Az Osztasok szubrutinban az sngAf ≠ 0 és sngBf ≠ 0 feltételek vizsgálatát is el kell végezni. A függvény Exit Function parancsához hasonlóan ebben az esetben az Exit Sub utasítás szolgál a főprogramba történő visszatérésre.

3.2.4.2 Reakció feladat

Írja meg a „Reakcio1” nevű programot, amely kiszámolja az „A → B” reakció lejátszódása alatt az „A” anyag koncentrációját [A].

A számolást kétféleképpen végezzük el, egyrészt a $\Delta[A] = -k \cdot [A] \cdot \Delta t$ (Euler féle) közelítéssel, mely szerint az „A” anyag koncentrációjának Δt időintervallum alatti változása arányos az „A” anyagnak a Δt időintervallum elején mért koncentrációjával. Az összefüggésben az arányossági tényező, a „k” reakciósebességi állandó értéke legyen 0,005 1/s)

	A	B	C	D
1	k=	0,005	1/s	
2	Idő	A=A-k*A*dt	A=A0*exp(-k*t)	Hiba
3	s	M	M	%
4	0	0,2	0,2	0
5	20	0,1800000	0,1809675	0,535
6	40	0,1620000	0,1637462	1,066
7	60	0,1458000	0,1481636	1,595
8	80	0,1312200	0,1340640	2,121
9	100	0,1180980	0,1213061	2,645
10	120	0,1062882	0,1097623	3,165
11	140	0,0956594	0,0993171	3,683
12	160	0,0860934	0,0898658	4,198
13	180	0,0774841	0,0813139	4,710
14	200	0,0697357	0,0735759	5,219
15	220	0,0627621	0,0665742	5,726
16	240	0,0564859	0,0602388	6,230
17	260	0,0508373	0,0545064	6,731
18	280	0,0457536	0,0493194	7,230
19	300	0,0411782	0,0446260	7,726

A másik számolás során a pontos, $[A]_t = [A]_0 \cdot e^{-k \cdot t}$ összefüggést használjuk, ahol [A]0 az „A” anyag t = 0 időpontban mért koncentrációja, [A]t az „A” anyagnak a reakció elindítása utáni „t” időpontban mért koncentrációja.

[A] értékét az 1. módszer esetén a Δ[A] koncentrációváltozást számoló EULER nevű, Double típusú Function hívásával számolja , melynek bemenő paraméterei: A ([A] az időintervallum elején), k és dt (Δt).

Számítsa ki az Euler féle közelítés relatív hibáját is:

$$\text{Hiba}\,(\%) = 100\% \times \frac{\left|[A]_{\text{Euler}} - [A]_{\text{exp}}\right|}{[A]_{\text{exp}}}$$

Itt $[A]_{Euler}$ az Euler közelítéssel, $[A]_{exp}$ az exponenciális kifejezéssel számolt érték. A program az InputBox paranccsal az alábbi értékeket kérje be: k (reakciósebességi állandó), tmax (a vizsgált

időtartomány), n (az időintervallumok száma a vizsgált időtartományban) A0 (az A anyag kezdeti koncentrációja.

A programot futtassa az „F2n15” munkalapon k=0,005, tmax=300, n=15, A0=0,2 bemenő adatokkal és az „F2n300” munkalapon való futtatáshoz k, tmax és A0 bemenő adatok legyenek az előző futtatás megfelelő adataival egyenlőek, n értéke pedig legyen 300. Hasonlítsa össze az eredményeket!

Megoldás:

A változók deklarálása után az InputBox parancs alkalmazásával a felhasználótól be kell kérni a reakciósebesség, a vizsgált időtartomány, az időintervallumok számának, valamint az anyag kezdeti koncentrációjának értékét és ki kell tölteni a táblázat fejlécét. A vizsgált időtartomány és az időintervallumok számának ismeretében az időintervallum számítható.

```
Sub Reakcio1()
    Dim i%, n%, k#, tmax#, dt#, t#, A0#, Akoz#, Apont#, hiba#
    k = InputBox(„k=?”, „Adatbekeres”, 0.005)
    Cells(1, 1) = „k=„: Cells(1, 2) = k: Cells(1, 3) = „1/s”
    tmax = InputBox(„tmax?”, , 300): n = InputBox(„n?”, , 15)
    dt = tmax / n
    A0 = InputBox(„A0?”, , 0.2)
        Cells(2, 1) = „Idő”: Cells(3, 1) = „s”
        Cells(2, 2) = „A=A-k*A*dt”: Cells(3, 2) = „M”
        Cells(2, 3) = „A=A0*exp(-k*t)”: Cells(3, 3) = „M”
        Cells(2, 4) = „Hiba”: Cells(3, 4) = „%”
```

A következő lépésben megadható a t = 0 időpillanathoz tartozó kezdeti koncentráció értéke. Bár az alábbiakban az Apont értékét az Expo1 függvény segítségével számolja ki a program, de az `Apont = 0.2` paranccsal is megadható. A hiba természetesen 0%. A táblázatba kiírandó értékeket a későbbiekben alkalmazandó i változó segítségével írja ki a program, de a `Cells(4, 1) = t`, `Cells(4, 2) = Akoz`, … parancsok is alkalmazhatóak.

```
i = 4: t = 0: Akoz = A0
Apont = Expo1(A0, k, t): hiba = 100 * Abs(Akoz - Apont) / Apont
    Cells(i, 1) = t: Cells(i, 2) = Akoz
    Cells(i, 3) = Apont: Cells(i, 4) = hiba
```

Az egyes időintervallumok utáni Akoz és Apont koncentrációértékek For - Next ciklus alkalmazásával és az Euler és Expo1 függvények segítségével számolhatóak. Az Euler függvény egyik bemenő értéke az aktuális Akoz értéke, melyre a $\Delta[A] = -k \cdot [A] \cdot \Delta t$ összefüggés miatt van szükség. A ciklusban i változó értékét meg kell növelni, hogy a számolt értékek kiírása a következő sorba történjen.

```
For t = dt To tmax Step dt
    Akoz = Euler(Akoz, k, dt)
    Apont = Expo1(A0, k, t)
    hiba = 100 * Abs(Akoz - Apont) / Apont
    i = i + 1
    Cells(i, 1) = t: Cells(i, 2) = Akoz
```

```
    Cells(i, 3) = Apont: Cells(i, 4) = hiba
Next t
End Sub
```

Végezetül az Euler és Expo1 függvényeket kell megírni. Az Euler függvény az első lépésben a koncentrációváltozás, a második lépésben az új koncentráció értékét számítja ki.

```
Function Euler(A#, k#, dt#) As Double
    Dim dA#
    dA = -A * k * dt
    Euler = A + dA
End Function
Function Expo1(A0#, k#, t#) As Double
    Expo1 = A0 * Exp(-k * t)
End Function
```

A programot n = 300 értékkel lefuttatva a tmax = 300 időpontban a hiba értéke csak 0,3755%, melynek oka, hogy az egyes időintervallumok huszadakkorák, mint n = 15 esetben, így az Euler féle közelítés közelebb van a valósághoz.

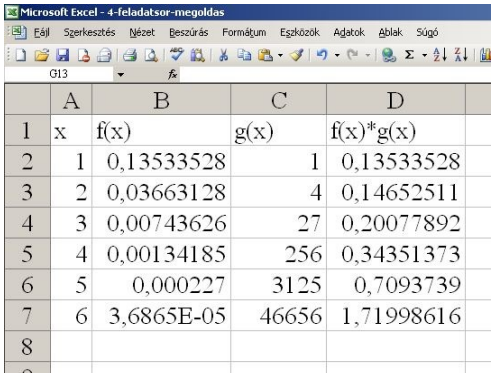
3.2.5 Fájlműveletek gyakorlatai

3.2.5.1 Kétfüggvényes feladat

Írja meg a „Szamolas” VBA programot, amely az Excel munkalap első sorába beírja a táblázat fejlécét, az „A” oszlopba Do-Loop ciklus alkalmazásával (ciklusváltozó: intI) beolvassa a „4-be.txt” fájl adatait (1, 2, 3, 4, 5, 6) az intBe változóba, majd függvények (Function) alkalmazásával kiszámítja a

$f(x) = x \cdot e^{-2x}$ és $g(x) = x^x$

függvények értékét („B” és „C” oszlop). Utána a „B” és „C” oszlop megfelelő elemeit összeszorozva töltse ki „D” oszlop celláit. Végezetül For-Next ciklusok alkalmazásával (ciklusváltozó: intI, intJ) az összes adatot írja ki a „4-ki.txt” file-ba. (intI, intJ intBe és intVeg is integer típusú változó) Az f(x) és g(x) értékét számoló függvények milyen típusúak legyenek?



	A	B	C	D
1	x	f(x)	g(x)	f(x)*g(x)
2	1	0,13533528	1	0,13533528
3	2	0,03663128	4	0,14652511
4	3	0,00743626	27	0,20077892
5	4	0,00134185	256	0,34351373
6	5	0,000227	3125	0,7093739
7	6	3,6865E-05	46656	1,71998616
8				

Megoldás:

A változók deklarálása után elkészíthető a táblázat fejléce, majd az intI változónak kell értéket adni. Az intI változóra a Do - Loop cikluson belül a f(x), g(x) és f(x) * g(x) értékek kiírásakor mint sorváltozóra lesz szükség. Mivel a cikluson belül intI változó értéke eggyel fog növekedni és az első értékeket a második sorba kell kiírni, így intI = 1 értéket célszerű adni.

```
Sub Szamolas()
    Dim intI As Integer, intJ%, intBe%, intVeg%
    Cells(1, 1) = „x”: Cells(1, 2) = „f(x)”: Cells(1, 3) = „g(x)”: Cells(1, 4) = „f(x)*g(x)”
    intI = 1
```

A következő lépésben a 4-be.txt fájlt kell megnyitni olvasásra és a Do - Loop ciklus segítségével kell be kell olvasni a fájl tartalmát. Javasolt elől tesztelő ciklust alkalmazni, mert „üres” 4-be.txt fájl esetében nincs beolvasandó adat és hátul tesztelő ciklust használva hibaüzenet jelenik meg. Általában nem ismert, hogy a fájlban mennyi adat van, ezért a beolvasást a file végéig (EOF: end of file, 1 a fájlazonosító) kell végezni:

```
Open „4-be.txt” For Input As #1
Do While Not EOF(1)
    intI = intI + 1
    Input #1, intBe
    Cells(intI, 1) = intBe
    intVeg = intI
Loop
Close #1
```

Az intVeg = intI utasítás biztosítja, hogy program elraktározza az utolsó, még adatokat tartalmazó sor számát. A következő lépésben az Exponencialis és Hatvany függvények segítségével és az intI ciklusváltozó értékét változtatva kiszámolhatóak f(x), g(x) és f(x) * g(x) értékek:

```
For intI = 2 To intVeg
    Cells(intI, 2) = Exponencialis(Cells(intI, 1))
    Cells(intI, 3) = Hatvany(Cells(intI, 1))
    Cells(intI, 4) = Cells(intI, 2) * Cells(intI, 3)
Next intI
```

Az adatbeolvasás, valamint f(x), g(x) és f(x) * g(x) értékének kiszámítása összevonható, ahogy az alábbi programrészletet mutatja:

```
Do While Not EOF(1)
    intI = intI + 1
    intVeg = intI
    Input #1, intBe
    Cells(intI, 1) = intBe
    Cells(intI, 2) = Exponencialis(intBe)
    Cells(intI, 3) = Hatvany(intBe)
    Cells(intI, 4) = Cells(intI, 2) * Cells(intI, 3)
Loop
```

Ebben az esetben a beolvasáskor nincs szükség az intVeg változó alkalmazására, de az adatok kiírásakor igen. Az adatok kiírásakor For - Next ciklus is alkalmazható, mert ismert a kiírandó adatok száma:

```
Open „4-ki.txt” For Output As #2
For intI = 1 To intVeg
    For intJ = 1 To 4
        If intJ < 4 Then
            Write #2, Cells(intI, intJ),
            Else
            Write #2, Cells(intI, intJ)
        End If
    Next intJ
Next intI
Close #2
End Sub
```

Bár a „Write #2, Cells(intI, intJ),” és „Write #2, Cells(intI, intJ)” parancsok között csak egy vessző az eltérés, ami azt okozza, hogy előbbi esetben a következő érték kiírása is ugyanabba a sorba történik, míg utóbbi esetben az érték kiírása után a program sortörést hajt végre. Végezetül a főprogram által használt függvényeket kell megírni.

```
Function Exponencialis(intBef!) As Single
    Exponencialis = intBef * Exp(-2 * intBef)
End Function

Function Hatvany(intBef%) As Long
    Hatvany = intBef ^ intBef
End Function
```

A hatványra emelés miatt a Hatvany függvény értéktartományának jóval nagyobbnak kell lennie, mint az intBef változó integer értéktartománya. Ennek a követelménynek megfelel a long adattípus.

3.2.6 Gyakorlatok tömbök alkalmazására

3.2.6.1 Prímszámok feladat

Írjon VBA programot, amelyben InputBox utasítással beolvas egy integer típusú számot (intN), majd a program keresse meg az összes 2 és intN közé eső prímszámot. A talált prímszámokat a Prim() tömb tárolja, majd a tömb tartalmát mentse ki a Primek-intN-ki.txt (pl: Primek-100-ki.txt) file-ba.

Megoldás:

A prímek meghatározása az ún. „eratosthenesi szita” segítségével történhet. Ennek lényege, hogy egy tömbbe beleírják a számokat intN-ig, majd kiveszik az összes számot, ami kettővel osztható. Ezután a tömbben maradt legkisebb szám a 3, ezért a tömbből kiveszik az az összes, hárommal osztható számot és így kell a továbbiakban is folytatni. A legnagyobb lehetséges prímszám intN négyzetgyöke lehet (pl: 529 esetében 23), így a szitálást intN négyzetgyökének egész részének elérésekor kell befejezni.

A változók deklarálása után be kell kérni intN értékét:

```
Sub Primek2()
    Dim Prim() As Integer, Munka%()
    Dim intIndex As Integer, intJ%, intK%, intN%, intVeg%
    intN = InputBox(„Kerem, adja meg intN erteket! (intN <= 32767)”, „Adatbekeres”, 400)
```

Azután meg kell határozni azt a számot, ahol a szitálást be kell fejezni (Sqr függvény négyzetgyököt von, Int függvény a szám egész részét veszi):

```
    intVeg = Int(Sqr(intN))
```

Következő lépés a számokat tartalmazó Munka tömb feltöltése:

```
    ReDim Munka(intN - 1)
    For intJ = 1 To intN - 1
        Munka(intJ) = intJ + 1
```

```
    Next intJ
```

Mivel a tömb első elemének értéke 2, ezért Munka(intN-1) = intN. A lehetséges prímszámokat a tömb a Munka(intVeg-1) = intVeg értékig tartalmazza, ezekkel a számokkal kell a tömbben lévő számokat megvizsgálni, melyre For - Next ciklus is használható (intK ciklusváltozó):

```
For intK = 1 To intVeg - 1
    If Munka(intK) > 0 Then
        For intJ = intK + 1 To intN - 1
            If Munka(intJ) > 0 Then
                If Munka(intJ) Mod Munka(intK) = 0 Then
                    Munka(intJ) = 0
                End If
            End If
        Next intJ
    End If
Next intK
```

A Munka(intK) > 0 kitétel azért szükséges, nehogy nullával osszunk, mert ha eredetileg Munka(intK) nem prímszám volt, akkor egy korábbi vizsgálat már lenullázta. A prímszámvizsgálat nyilvánvalóan a tömb következő (intK+1) elemétől kezdődhet, erre szintén egy For - Next ciklus használható (intJ ciklusváltozó). Az Munka(intJ) > 0 kitétel azért kell, hogy a kinullázott számokat már ne kelljen vizsgálni. Amennyiben a maradékosztás (Mod parancs) eredménye nulla, Munka(intJ) nem prímszám, értékét le kell nullázni. A következő lépésben a Munka tömb nullától eltérő értékeit kell átmásolni a Prim tömbbe:

```
intIndex = 0
For intJ = 1 To intN - 1
If Munka(intJ) > 0 Then
    intIndex = intIndex + 1
    ReDim Preserve Prim(intIndex)
    Prim(intIndex) = Munka(intJ)
End If
Next intJ
```

Az átmásolás után a Prim tömb elemeinek számát intIndex tartalmazza. A fájlba kiírás történhet For - Next vagy Do - Loop ciklussal is. Do - Loop ciklus alkalmazása esetén a ciklusváltozó értékét minden ciklusban meg kell növelni.

```
Open „Primek2-” & intN & „-ki.txt” For Output As #1
    intJ = 0
    Do
        intJ = intJ + 1
        Write #1, Prim(intJ)
    Loop While intJ < intIndex
Close #1
End Sub
```

3.2.6.2 Legnagyobb közös osztó és legkisebb többszörös feladat

Írjon VBA programot, amely először az intPrim tömbbe egymást követő prímszámokat olvas be (első eleme 2), majd kiszámítja az Inputbox paranccsal beadott intSzam1 és intSzam2 (integer típusú) számok legkisebb közös többszörösét (lngLKT, long) és legkisebb közös osztóját (intLKO,

integer). Az intTomb (integer) tömb egyes oszlopaiban tárolja intSzam1, intSzam2, lngLKT és intLKO törzstényezős alakjukban szereplő prímszámok hatványkitevőit. A MsgBox utasítással írassa ki intSzam1, intSzam2, lngLKT és intLKO értékét és mentse le a LKT_LKO_intSzam1_intSzam2-ki.txt (pl: LKT_LKO_128_162-ki.txt) fájlba intSzam1, intSzam2, lngLKT és intLKO értékét, valamint intPrim és intTomb tömböket.

Megoldás:

A legnagyobb közös osztó és legkisebb közös többszörös meghatározása során először a felhasználó által megadott számok törzstényezős alakját kell meghatározni (például $24 = 2^3 \times 3$). Ez úgy történhet, hogy a számokat prímszámokkal kell osztani és az így kapott értéket az intMaradek1 és intMaradek2 változókban kell tárolni és a prímszámokkal való osztást addig kell végezni, míg intMaradek1 > 1 és intMaradek2 > 1.

A változók deklarálásakor figyelni kell arra, hogy a legkisebb közös többszörös értéke nagyobb, mint a felhasználó által megadott számok értéke (amennyiben a mindkét szám prímszám, akkor a legkisebb közös többszörös értéke a két szám szorzata), ezért a legkisebb közös többszörös értékét tartalmazó változó gyakran nem lehet integer típusú.

```
Sub LKT_LKO()
    Dim intPrim() As Integer, intTomb%()
    Dim intI As Integer, intJ%, intLKO%, intMaradek1%, intMaradek2%
    Dim intPrimMax%, intPrimVeg%, intSzam1%, intSzam2%
    Dim lngLKT As Long
    Dim strFnev As String
```

Ezután meg kell kérni a felhasználót, hogy nyissa mega prímszámokat tartalmazó fájlt:

```
MsgBox „Kerem, nyissa meg a primszamokat tartalmazo file-t.”
strFnev = Application.GetOpenFilename
Open strFnev For Input As #1
```

Nem ismert, hogy a fájl mennyi prímszámot tartalmaz, ezért az adatok beolvasását a fájl végéig és Do - Loop ciklus alkalmazásával kell végezni. Az intPrim dinamikus tömb méretét minden beolvasáskor meg kell növelni a Redim Preserve paranccsal (A Redim önmagában nem elég, mert akkor a tömb korábbi elemeinek értékét lanulláza a program):

```
intI = 0
Do While Not EOF(1)
   intI = intI + 1
   ReDim Preserve intPrim(intI)
   Input #1, intPrim(intI)
Loop
Close #1
```

Az intI változó megadja, hogy hány prímszámot olvasott be a program, értékét célszerű átadni az intPrimVeg változónak, mert intI ciklusváltozó, így értéke később átírásra kerülhet.

```
intPrimVeg = intI
```

Ezt követően definiálni kell intTomb méretét és a tömböt fel kell tölteni nulla értékekkel. A tömb egyik dimenziója a beolvasott prímszámok darabszáma, míg a másik dimenzió 4 (intSzam1, intSzam2, lngLKT és intLKO törzstényezős alakjainak hatványkitevőinek tárolására)

```
ReDim intTomb(intPrimVeg, 4)
For intI = 1 To intPrimVeg
    For intJ = 1 To 4
    intTomb(intI, intJ) = 0
    Next intJ
Next intI
```

Ezután a felhasználótól bekérhető a két szám, de arra ügyelni kell, hogy a számok nem lehetnek nagyobbak a legnagyobb beolvasott prímszám négyzeténél (lásd Prímszámok feladat):

```
Do
    intSzam1 = InputBox(„Kerem, adja meg Szam1 erteket (< 32767)!", „Adatbekeres")
    intSzam2 = InputBox(„Kerem, adja meg Szam2 erteket (< 32767)!", „Adatbekeres")
    If intSzam1 > intPrim(intPrimVeg) ^ 2 Or intSzam2 > intPrim(intPrimVeg) ^ 2 Then
        MsgBox „Szam1 és/vagy Szam2 > a beolvasott legnagyobb prímszám négyzete"
    End If
Loop While intSzam1 > intPrim(intPrimVeg) ^ 2 Or intSzam2 > intPrim(intPrimVeg) ^ 2
```

A beolvasott számok értékét át kell adni intMaradek1 és intMaradek2 változóknak:

```
intMaradek1 = intSzam1
intMaradek2 = intSzam2
```

Utána kezdődhet intMaradek1 és intMaradek2 változók prímszámokkal való osztása. Nem lehet előre tudni, hogy a beolvasott prímszámok közül mennyit kell felhasználni az osztások során, ezért Do - Loop ciklust kell alkalmazni:

```
intI = 1
Do While intMaradek1 > 1 Or intMaradek2 > 1
    If intMaradek1 Mod intPrim(intI) = 0 Or intMaradek2 Mod intPrim(intI) = 0 Then
        If intMaradek1 Mod intPrim(intI) = 0 Then
            intMaradek1 = intMaradek1 / intPrim(intI)
            intTomb(intI, 1) = intTomb(intI, 1) + 1
        Else
            intMaradek2 = intMaradek2 / intPrim(intI)
            intTomb(intI, 2) = intTomb(intI, 2) + 1
        End If
        Else
            intI = intI + 1
    End If
Loop
```

A külső If … then - End If szerkezet megvizsgálja, hogy intMaradek1 vagy intMaradek2 osztható-e az adott prímszámmal. Amennyiben igen, akkor a belső if … then - End If szerkezetével megnöveli eggyel intTomb megfelelő elemének az értékét, míg ellentétes esetben a ciklusváltozó értékét növeli meg. Az egymásba ágyazott If … then - End If szerkezetek lehetővé teszik, hogy ugyanazzal a prímszámmal többször is osztani lehessen. Természetesen az alábbi szerkezet is alkalmazható:

```
intI = 1
Do While intMaradek1 > 1 Or intMaradek2 > 1
    If intMaradek1 Mod intPrim(intI) = 0 Or intMaradek2 Mod intPrim(intI) = 0 Then
        If intMaradek1 Mod intPrim(intI) = 0 Then
```

```
            intMaradek1 = intMaradek1 / intPrim(intI)
            intTomb(intI, 1) = intTomb(intI, 1) + 1
        End If
        If intMaradek2 Mod intPrim(intI) = 0 Then
            intMaradek2 = intMaradek2 / intPrim(intI)
            intTomb(intI, 2) = intTomb(intI, 2) + 1
        End If
        Else
        intI = intI + 1
    End If
Loop
```

Az intI változó most azt adja meg, hogy az intPrim tömbbe beolvasott prímszámok közül mennyit használt fel a program addíg, amíg intMaradek1 = 1 és intMaradek2 = 1 lett, értékét célszerű átadni az intPrimMax változónak.

```
intPrimMax = intI
```

A törzstényezős alakok hatványkitevőinek meghatározása után kiindulási értéket kell adni intLKO és lngLKT változóknak, valamint intI ciklusváltozónak. Mivel intLKO és lngLKT értékét szorzással lehet megkapni a kiindulási érték csak 1 lehet.

```
intLKO = 1
lngLKT = 1
intI = 1
```

Az intTomb változóban lévő értékek segítségével megadható intLKO és lngLKT hatványkitevői (a kisebb érték intLKO, a nagyobb lngLKT értékének meghatározásához szükséges). Mivel intPrimMax értéke ismert, ezért mind Do - Loop ciklus, mind For - Next ciklus alkalmazható:

```
Do
    If intTomb(intI, 1) > intTomb(intI, 2) Then
        intTomb(intI, 3) = intTomb(intI, 1)
        intTomb(intI, 4) = intTomb(intI, 2)
        Else
        intTomb(intI, 3) = intTomb(intI, 2)
        intTomb(intI, 4) = intTomb(intI, 1)
    End If
    lngLKT = lngLKT * intPrim(intI) ^ intTomb(intI, 3)
    intLKO = intLKO * intPrim(intI) ^ intTomb(intI, 4)
    intI = intI + 1
Loop While intI <= intPrimMax
```

Ezt követi intSzam1, intSzam2, lngLKT és intLKO értékének képernyőre és fájlba írása:

```
MsgBox „Szam1: „ & intSzam1 & „, „ & „Szam2: „ & intSzam2 & „, „ & „LKT: „ & lngLKT & „, „ &
„LKO: „ & intLKO
Open „LKT_LKO_” & intSzam1 & „_” & intSzam2 & „-ki.txt” For Output As #2
Write #2, „Primek”, „Szam1”, „Szam2”, „LKT”, „LKO”
Write #2, „---”, intSzam1, intSzam2, lngLKT, intLKO
```

A program a fájlba azt is beleírja, hogy a fenti négy változó, mely prímszámok (intPrim(intI)) hányadik hatványát (intTomb(intI,1-től 4-ig)) tartalmazzák törzstényezős szorzatukban:

```
intI = 1
Do While intI <= intPrimMax
    Write #2, intPrim(intI),
```

```
    Write #2, intTomb(intI, 1), intTomb(intI, 2), intTomb(intI, 3), intTomb(intI, 4)
    intI = intI + 1
Loop
Close #2
End Sub
```

3.2.6.3 Térfogatszámolás feladat

A korábban megírt programot úgy kell módosítani, hogy ciklusokat, tömböket, függvényeket és a szubrutint is tartalmazzon, ezért az alábbiakat kell figyelembe venni:

- R, V, A és f értékeket lehet tömbökben tárolni, míg strDontes és sngPi változókat nem célszerű (lehet 1 elemes tömböket deklarálni, de nincs értelme),
- az if then - goto ujra szerkezetek helyett ciklusokat kell alkalmazni. Mivel nem lehet tudni, hogy a ciklusnak hányszor kell lefutnia, ezért Do - Loop ciklust kell használni. A blokkdiagramon látható, hogy a feltételek ($R_1>R_2$, új számolás) vizsgálata a ciklus végén történik, ezért hátul tesztelő ciklusra van szükség,
- R, V, A és f értékeket tartalmazó tömbök elemszáma ismert (elemszám: 2, 3, 3, 3), így a tömbökkel végzett műveleteknél For - Next ciklust is lehet használni, melyhez viszont az „i” ciklusváltozó deklarálása szükséges.

A fentieket figyelembe véve a változók deklarálása és π értékének kiszámolása a következőképpen történhet:

```
Sub Terfogatszamolas2()
    Dim sngR!(2), sngV!(3), sngA!(3), sngf!(3)
    Dim i%
    Dim strDontes As String
    Dim sngPi As Single
    sngPi = 4 * Atn(1)
```

Ezt követi a két Do - Loop ciklus megnyitása és a két sugárérték beolvasása:

```
Do
    Do
        sngR(1) = InputBox(„R1 = ?”, „Adatbekeres”, 2)
        sngR(2) = InputBox(„R2 = ? (kisebb legyen, mint R1)”, „Adatbekeres”, 1)
```

Bár For - Next ciklussal is megoldható az adatok beolvasása, de nem célszerű, mert a felhasználónak kiírandó szöveg jelentősen eltér, ami miatt if - else - End If szerkezetet kellene használni, de ez bonyolultabbá tenné a programot. A következő lépés a $R_1>R_2$ feltétel vizsgálata, az első ciklus bezárása:

```
    Loop While sngR(2) >= sngR(1)
```

Ezután a térfogat- és felületértékek számítása következik:

```
For i = 1 To 2
    sngV(i) = Terfogat(sngR(i), sngPi)
    sngA(i) = Felulet(sngR(i), sngPi)
Next i
sngV(3) = sngV(1) - sngV(2)
sngA(3) = sngA(1) + sngA(2)
```

A térfogat, illetve a felület számolásához a „Terfogat”, illetve „Felulet” függvények használhatóak, melyeknek a program átadja a sugár és π értékét. Ez a programrész azonban eltér a blokkdiagramon láthatótól. A blokkdiagram szerint a számolásnak a V_1, V_2, V_3, A_1, A_2, A_3 sorrendben kellene történnie, míg a fenti programrészlet V_1, A_1, V_2, A_2, V_3, A_3 sorrendet alkalmaz. A blokkdiagramnak az alábbi programrészlet felelne meg:

```
For i = 1 To 2
    sngV(i) = Terfogat(sngR(i), sngPi)
Next i
sngV(3) = sngV(1) - sngV(2)
For i = 1 To 2
    sngA(i) = Felulet(sngR(i), sngPi)
Next i
sngA(3) = sngA(1) + sngA(2)
```

Ebben az esetben viszont két For - Next ciklust kell alkalmazni, de az első megoldással az egyik For - Next ciklus elhagyható. Most következik a fajlagos felület értékek kiszámolása és az eredmények kiírása:

```
For i = 1 To 3
    sngf(i) = sngA(i) / sngV(i)
    MsgBox „V” & i & „ = „ & sngV(i) & „, A” & i & „ = „ & sngA(i) & „, f” & i & „= „ & sngf(i)
Next i
```

A blokkdiagram f_1, f_2, f_3, Kiír1, Kiír2, Kiír3 sorrendet, míg az előző programrészlet f_1, Kiír1, f_2, Kiír2, f_3, Kiír3 sorrendet alkalmaz. Végül az „Uj szamolas” feltétel vizsgálata, a második ciklus bezárása és a program vége következik:

```
strDontes = InputBox(„Szeretne uj szamolast vegezni? (i = igen, bármi más = nem)”, „Adatbekeres”, „i”)
Loop While strDontes = „i”
MsgBox „Program vege”
End Sub
```

A „Terfogat” és „Felulet” függvények sngRbe és sngPibe változóiba adja át a főprogram sngR(1), sngR(2) és sngPi értékét. Nem okoz gondot, hogy mindkét függvény egy-egy sngRbe és sngPibe változót használ, mert az azonos nevű változók alkalmazási területe nem fed át.

```
Function Terfogat(sngRbe!, sngPibe!) As Single
    Terfogat = 4 * sngRbe ^ 3 * sngPibe / 3
End Function
Function Felulet(sngRbe!, sngPibe!) As Single
    Felulet = 4 * sngRbe ^ 2 * sngPibe
End Function
```

Ha a két függvény helyett egy szubrutint használ a program, akkor a főprogramban a

```
For i = 1 To 2
    sngV(i) = Terfogat(sngR(i), sngPi)
    sngA(i) = Felulet(sngR(i), sngPi)
Next i
```

programrészlet helyett a

```
For i = 1 To 2
    Call Terfogat_Felulet(sngR(i), sngPi, sngV(i), sngA(i))
```

```
Next i
```

programrészletet kell alkalmazni, míg maga a „Terfogat_Felulet” szubrutin a következőképpen néz ki:

```
Sub Terfogat_Felulet(sngRbe!, sngPibe!, sngVki!, sngAki!)
    sngVki = 4 * sngRbe ^ 3 * sngPibe / 3
    sngAki = 4 * sngRbe ^ 2 * sngPibe
End Sub
```

A szubrutin az sngVki és sngAki változók értékét adja át a főprogram sngV(i) és sngA(i) változóinak.

3.2.6.4 Öröknaptár feladat

A korábban megírt programot úgy kell változtatni, hogy a Select Case - End Select utasítást kell kiváltani egy tömbbel, mely a hét napjait tartalmazza.

```
Sub Oroknaptar_tomb()
    Dim datReferencia As Date, datKerdesesNap As Date
    Dim intMaradek As Integer
    Dim sngKulonbseg As Single
    Dim Napok(6) As String
```

A Napok tömb elemeit fel kell tölteni a hét napjaival. Mivel a maradékosztás eredménye nulla is lehet, ezért a napokat célszerű a tömb nulladik elemétől a hatodik elemig beadni.

```
Napok(0) = „hetfo”
Napok(1) = „kedd”
Napok(2) = „szerda”
Napok(3) = „csutortok”
Napok(4) = „pentek”
Napok(5) = „szombat”
Napok(6) = „vasarnap”
datReferencia = #8/31/2009#
datKerdesesNap = InputBox(„Kerem, adja meg a kerdeses nap datumat!”, „Adatbekeres”, #1/1/2009#)
sngKulonbseg = datKerdesesNap - datReferencia
intMaradek = sngKulonbseg Mod 7
```

Amennyiben intMaradek értéke negatív szám, akkor a változó értékét héttel meg kell növelni. Ezután következhet a kérdéses nap kiírása.

```
If intMaradek < 0 Then intMaradek = intMaradek + 7
MsgBox „A kerdeses nap „ & Napok(intMaradek) & „ volt/lesz.”
End Sub
```

Másik lehetséges megoldás, amikor a variant típusú Napok tömb elemeit felsorolásszerűen adják meg:

```
Dim Napok As Variant
Napok = Array(„hetfo”, „kedd”, „szerda”, „csutortok”, „pentek”, „szombat”, „vasarnap”)
```

A program többi része megegyezik a fentebb leírtakkal.

Egyenletmegoldás feladat

Írja meg az „Egyenletmegoldas” nevű VBA programot, amely először beolvassa az alábbi két egyenletrendszer közül az egyik együtthatóit az „Egyenletek.txt” vagy az „Egyenletek2.txt” file-ból egy 3×4 nagyságú tömbbe, majd az egyenletrendszert megoldja és a tömb tartalmát egyrészt a munkalap A1:D3 celláiba, másrészt az „Egyenletek-megoldas.txt” file-ba írja ki. A program írásakor az sngHanyados, sngTomb(3,4) (mindkettő single), az intI, intJ, intK (mind integer) változókat alkalmazza.

$$6x+2y-3z=-4 \quad -x+3y+4z=-2 \quad 2x-3y-3z=9$$

$$2x+-6y-8z=4 \quad 6x+2y-3z=-4 \quad 2x-3y-3z=9$$

```
6,2,-3,-4
-1,3,4,-2
2,-3,-3,9
```

	A	B	C	D
1	1	0	0	3
2	0	1	0	-5
3	0	0	1	4

```
1,0,0,3
0,1,0,-5
0,0,1,4
```

egyenletrendszerek Egyenletek.txt megoldás Excel munkalapon és fájlban

Megoldás:

A változók deklarálása után meg kell nyitni az Egyenletek.txt fájlt, majd a fájlban található értékekkel fel kell tölteni az sngTomb tömböt, melyre két egymásba ágyazott For - Next ciklust kell használni.

```
Sub Egyenletmegoldas()
    Dim intI As Integer, intJ%, intK%
    Dim sngHanyados As Single
    Dim sngTomb(3, 4) As Single
    Open „Egyenletek.txt” For Input As #1
        For intI = 1 To 3
            For intJ = 1 To 4
                Input #1, sngTomb(intI, intJ)
                Cells(intI, intJ) = sngTomb(intI, intJ)
        Next intJ
    Next intI
Close #1
```

	A	B	C	D
1	6	2	-3	-4
2	-1	3	4	-2
3	2	-3	-3	9
4				
5	1. lépés			
6	6	2	-3	-4
7	6	-18	-24	12
8	6	-9	-9	27
9				
10	2.lépés			
11	6	2	-3	-4
12	0	-20	-21	16
13	0	-11	-6	31
14				
15	3. lépés			
16	6	2	-3	-4
17	0	-20	-21	16
18	0	-20	-10,909	56,3636
19				
20	4. lépés			
21	6	2	-3	-4
22	0	-20	-21	16
23	0	0	10,0909	40,3636

Ezután az egyenlet együtthatóit tartalmazó tömbrész (3 × 3) diagonálisa alatti számokat kell kinullázni. Először az sngHanyados változó segítségével meg kell határozni, hogy a tömb egyes sorainak elemeit mekkora értékkel kell megszorozni a sorok egymásból való kivonása előtt. Az intI a kivonandó sor indexe, az intJ ahhoz a sorhoz tartozik, melyből a kivonás történik, míg intK az egyes sorok elemeihez tartozik (a jobboldalon lévő ábra a program működését mutatja, de ezt a program a működése során nem írja ki).

```
For intI = 1 To 2
    For intJ = intI + 1 To 3
        sngHanyados = sngTomb(intI, intI) / sngTomb(intJ, intI)
        For intK = 1 To 4
                sngTomb(intJ, intK) = sngHanyados * sngTomb(intJ, intK)
                sngTomb(intJ, intK) = sngTomb(intJ, intK) - sngTomb(intI, intK)
        Next intK
    Next intJ
Next intI
```

Ezt követi a diagonális feletti számok kinullázása. A számolás során mindenképp a lépésköz értéke -1 (Step -1), mert a magasabb számú soroktól kell visszafelé haladni az első sor felé.

```
For intI = 3 To 2 Step -1
    For intJ = intI - 1 To 1 Step -1
        sngHanyados = sngTomb(intI, intI) / sngTomb(intJ, intI)
        For intK = 1 To 4
                sngTomb(intJ, intK) = sngHanyados * sngTomb(intJ, intK)
                sngTomb(intJ, intK) = sngTomb(intJ, intK) - sngTomb(intI, intK)
        Next intK
    Next intJ
Next intI
```

Ezután a diagonális értékeit kell 1-re normálni.

```
For intI = 1 To 3
    sngHanyados = sngTomb(intI, intI)
    sngTomb(intI, intI) = sngTomb(intI, intI) / sngHanyados
    sngTomb(intI, 4) = sngTomb(intI, 4) / sngHanyados
Next intI
```

Végezetül a sngTomb tömb értékeit kell fájlba írni. Az If - End If szerkezet teszi lehetővé, hogy a következő kiírandó értéket ugyanabba vagy egy új sorba kell kiírni (van, illetve nincs vessző/pontosvessző a write parancs sorának végén).

	A	B	C	D
25	5.lépés			
26	-20,182	-6,7273	10,0909	13,4545
27	0	9,61039	10,0909	-7,6883
28	0	0	10,0909	40,3636
29				
30	6.lépés			
31	-20,182	-6,7273	0	-26,909
32	0	9,61039	0	-48,052
33	0	0	10,0909	40,3636
34				
35	7. lépés			
36	28,8312	9,61039	0	38,4416
37	0	9,61039	0	-48,052
38	0	0	10,0909	40,3636
39				
40	8.lépés			
41	28,8312	0	0	86,4935
42	0	9,61039	0	-48,052
43	0	0	10,0909	40,3636
44				
45	9. lépés			
46	1	0	0	3
47	0	1	0	-5
48	0	0	1	4

```
Open „Egyenletek-megoldas.txt” For Output As #2
For intI = 1 To 3
    For intK = 1 To 4
        Cells(intI, intK) = sngTomb(intI, intK)
        If intK < 4 Then
            Write #2, sngTomb(intI, intK);
        Else
            Write #2, sngTomb(intI, intK)
        End If
    Next intK
Next intI
Close #2
End Sub
```

3.2.7 Önállóan megoldandó gyakorló feladatok

3.2.7.1 For-Next ciklus

Szorzótábla feladat

Írassa ki a 12×12 szorzótáblát egy üres munkalapra.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1														
2			1	2	3	4	5	6	7	8	9	10	11	12
3			2	4	6	8	10	12	14	16	18	20	22	24
4			3	6	9	12	15	18	21	24	27	30	33	36
5			4	8	12	16	20	24	28	32	36	40	44	48
6			5	10	15	20	25	30	35	40	45	50	55	60
7			6	12	18	24	30	36	42	48	54	60	66	72
8			7	14	21	28	35	42	49	56	63	70	77	84
9			8	16	24	32	40	48	56	64	72	80	88	96
10			9	18	27	36	45	54	63	72	81	90	99	108
11			10	20	30	40	50	60	70	80	90	100	110	120
12			11	22	33	44	55	66	77	88	99	110	121	132
13			12	24	36	48	60	72	84	96	108	120	132	144

Írjon programot, ami a szorzótábla alsó háromszög részletét jeleníti meg! A sorok számát Inputbox utasítással kérdezze meg.

	1	2	3	4	5	6	7	8	9	10	11
1											
2											
3		1									
4		2	4								
5		3	6	9							
6		4	8	12	16						
7		5	10	15	20	25					
8		6	12	18	24	30	36				
9		7	14	21	28	35	42	49			
10		8	16	24	32	40	48	56	64		
11		9	18	27	36	45	54	63	72	81	
12		10	20	30	40	50	60	70	80	90	100

Pascal háromszög feladat

Írassa ki a Pascal háromszög egy részletét a munkalapra! Inputbox utasítással kérdezzen rá, mennyi sort írjon ki belőle a program! (http://hu.wikipedia.org/wiki/Pascal-h%C3%A1romsz%C3%B6g)

	1	2	3	4	5	6
1		1				
2		1	1			
3		1	2	1		
4		1	3	3	1	
5		1	4	6	4	1

Írassa ki a Pascal háromszög egy részletét a munkalapra! Inputbox utasítással kérdezzen rá, mennyi sort írjon ki belőle a program, de ezúttal az alábbi ábrán láthatóan, szépen elrendezve jelenjen meg a háromszög!

	1	2	3	4	5	6	7	8	9	10	11	12
1							1					
2						1		1				
3					1		2		1			
4				1		3		3		1		
5			1		4		6		4		1	
6		1		5		10		10		5		1

3.2.7.2 Do-Loop ciklus

Véletlen sorsolás

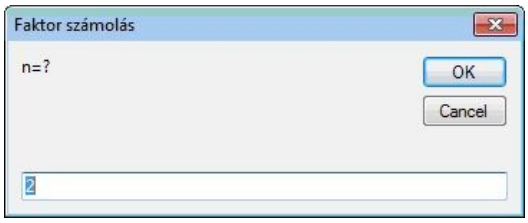
A program véletlen számokat sorsoljon ki és ezeket írja egymás alatti cellákba, amíg egy megadott határértéknél nagyobbat nem sorsol ki, ekkor hagyja abba a sorsolást és írja ki a kisorsolt számok összegét és átlagukat. A határértéket a sorsolás megkezdése előtt egy Inputbox utasítással kérdezze meg (az alapérték legyen 0.95)

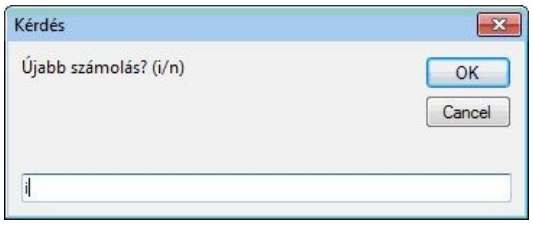
Véletlen számot az Rnd utasítással tud generálni (pl: X = rnd), mely utasítás [0,1] intervallumba eső véletlenszerű érteket add.

	A	B	C	D	E	F
1	n	x		N	összeg	átlag
2	1	0,910964		4	2,812949	0,703237
3	2	0,226866				
4	3	0,695116				
5	4	0,980003				
6	5	0,524868				
7	6	0,767112				
8	7	0,053505				
9	8	0,592458				
10	9	0,4687				
11	10	0,298165				
12	11	0,622697				
13	12	0,647821				
14	13	0,263793				
15	14	0,279342				
16	15	0,829802				
17	16	0,824602				
18	17	0,589163				
19	18	0,986093				

Faktoriális számítása

Írjon programot, ami egy szám faktoriálisát számolja ki! A számot Inputbox utasítással kérdezze meg a felhasználótól! A számolás után az eredményt a Msgbox utasítással írja ki a program. Végül a program kérdezze meg szeretnénk-e új számolást, ha a válasz „i”, akkor kezdje az egészet elölről.





3.2.7.3 Szubrutin alkalmazása

Oszthatóság vizsgálata

Töltsön fel egy 10×10 mezőt véletlen számokkal, melyek értéke 1-100 közé esik (Int(Rnd * 100 + 1). Majd a hárommal osztható számokat tartalmazó cellákat színezze pirosra, az öttel oszthatókat

kékre, a héttel oszthatókat zöldre. A színezéshez használjon szubrutint, amelynek a paraméterei: egy string (a szín megadása), két integer (a színezendő cella koordinátái).

Egy cella színét a következő utasítással változtathatja meg:

Cells(y, x).Interior.Color = RGB(0, 200, 0) zöld

Cells(y, x).Interior.Color = RGB(200, 0, 0) piros

Cells(y, x).Interior.Color = RGB(0, 0, 200) kék

	A	B	C	D	E	F	G	H	I	J
1	79	6	52	76	81	33	98	81	68	91
2	88	42	13	96	80	70	41	2	17	17
3	53	41	11	28	65	85	50	19	90	38
4	33	78	22	45	24	88	62	38	39	87
5	59	94	52	34	87	26	26	18	35	1
6	75	85	28	71	41	82	75	44	8	42
7	34	72	32	80	16	60	96	25	94	12
8	99	64	60	91	58	25	87	8	44	76
9	25	38	40	53	28	59	21	8	90	12
10	66	91	24	95	85	45	50	77	84	39

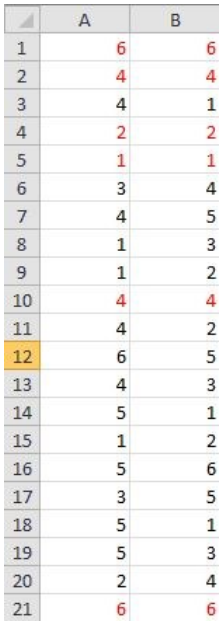
3.2.7.4 Összetett feladatok

Kockadobás feladat

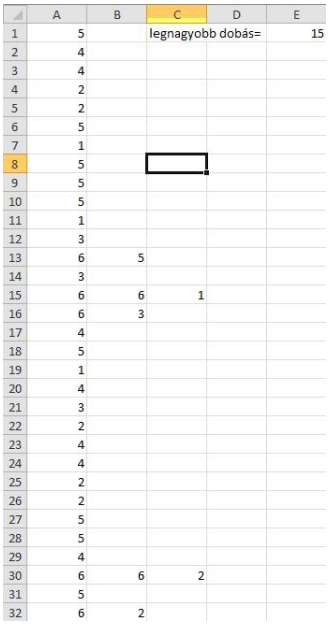
Dobjon a számítógép két hatoldalú kockával és a dobások eredményét két, egymás melletti oszlopba írja ki. Ezt addig ismételje, míg ötször nem dob azonosat a két kockával. Ezután a dobások számát írja ki egy külön ablakba (Msgbox utasítás)

A dobásokat külön függvénnyel (function) számolja ki. Véletlen egész számokat az Int((Rnd × 6) + 1) utasítással tud generálni.

A kettős dobásokat tartalmazó cellákban a szöveg színét a jobb láthatóság érdekében a következő utasítás megfelelő használatával pirosra változtathatja: Cells(x, y).Font.Color = RGB(255, 0, 0)



	A	B
1	6	6
2	4	4
3	4	1
4	2	2
5	1	1
6	3	4
7	4	5
8	1	3
9	1	2
10	4	4
11	4	2
12	6	5
13	4	3
14	5	1
15	1	2
16	5	6
17	3	5
18	5	1
19	5	3
20	2	4
21	6	6



	A	B	C	D	E
1	5		legnagyobb dobás=		15
2	4				
3	4				
4	2				
5	2				
6	5				
7	1				
8	5				
9	5				
10	5				
11	1				
12	3				
13	6	5			
14	3				
15	6	6	1		
16	6	3			
17	4				
18	5				
19	1				
20	4				
21	3				
22	2				
23	4				
24	4				
25	2				
26	2				
27	5				
28	5				
29	4				
30	6	6	2		
31	5				
32	6	2			

Kockadobás újra feladat

Dobjon a program hatoldalú kockával százszor, miközben a dobások értékét egymás alatti cellákba írja ki. Hatos dobás esetén dobjon újra és az értéket a mellette lévő cellába írja be, ismételt hatos dobás esetén az új számot a harmadik oszlopba, így tovább még végül nem hatost dob.

Az egy sorba írt számokat egy dobásnak véve (pl: két hatos és egy kettes dobás értéke: 14), keresse meg a legnagyobb dobást és a dobást tartalmazó sor számát írja ki.

Táblázat feladat

Egy táblázatba írasson ki véletlen egész számokat, a táblázat sorainak és oszlopainak számát Inputbox utasítással kérdezze meg. A kiírt véletlen számok maximális értéke legyen oszlop szám +1 (például a harmadik oszlopban 1, 2, 3, 4 a lehetséges értékek). Az első sorban a lehetséges maximumértékek szerepeljenek. A véletlen értékek számolásához írjon függvényt, amely bemenő értéke a lehetséges maximum.

A páros számokat színezze pirosra! (az Int utasítást használhatja)

	A	B	C	D	E	F	G	H	I	J
1	max=	3	4	5	6	7	8	9	10	11
2	1	3	3	3	3	4	1	2	5	5
3	1	1	3	3	3	3	1	7	5	4
4	2	3	4	2	1	1	3	5	5	10
5	1	2	4	3	3	6	2	4	3	9
6	2	1	1	2	2	4	1	2	4	4
7	1	3	3	1	4	6	5	7	1	8
8	1	2	4	2	4	5	6	9	6	3
9	2	2	2	5	1	6	5	5	2	1
10	1	2	3	1	6	1	7	1	10	1
11	1	1	3	2	4	2	1	7	1	4

Megoldások

For-Next ciklus

Szorzótábla feladat

Teljes szorzótábla kiíratása:

```
Sub szorzotábla()
Dim i As Integer
Dim k As Integer
For i = 1 To 12
    For k = 1 To 12
      Cells(k + 1, i + 2) = i * k
    Next k
Next i
End Sub
```

Alsó háromszög kiíratása:

```
Sub haromszog()
Dim i As Integer
Dim k As Integer
Dim m As Integer
```

```
m = InputBox(„Mennyi?”, „Kérdés”, 5)
For i = 1 To m
    For k = 1 To i
        Cells(i + 2, k + 1) = i * k
    Next k
Next i
End Sub
```

Pascal háromszög feladat

Pascal háromszög kiíratása:

```
Sub Pascal1()
Dim i%, k%, n%
n = InputBox(„Mennyi sort?”, „kérdés”, 4)
Cells(1, 2) = 1
For i = 2 To n + 1
    For k = 2 To i + 1
        Cells(i, k) = Cells(i - 1, k - 1) + Cells(i - 1, k)
    Next k
Next i
End Sub
```

Pascal háromszög szebb elrendezésű kiíratása:

```
Sub Pascal2()
Dim i%, k%, n%, x%
Cells.HorizontalAlignment = xlCenter
n = InputBox(„mennyi?”, „kérdés”, 4)
Cells(1, n + 1) = 1
For i = 2 To n
    For k = 1 To i
        x = k * 2 - i + n
        Cells(i, x) = Cells(i - 1, x - 1) + Cells(i - 1, x + 1)
    Next k
Next i
End Sub
```

Do-Loop ciklus

Véletlen sorsolás

```
Sub Veletlen()
    Dim x!, osszeg!, limit!
    Dim szamlalo%
    Cells(1, 1) = „n”
    Cells(1, 2) = „x”
    Cells(1, 4) = „N”
    Cells(1, 5) = „összeg”
    Cells(1, 6) = „átlag”
    limit = InputBox(„A határ érték (0 - 1)”, „Kérdés”, „0,95”)
    szamlalo = 0
    osszeg = 0
Do
    x = Rnd
    szamlalo = szamlalo + 1
    osszeg = osszeg + x
    Cells(szamlalo + 1, 1) = szamlalo
    Cells(szamlalo + 1, 2) = x
Loop Until x > limit
    Cells(2, 4) = szamlalo
```

```
    Cells(2, 5) = osszeg
    Cells(2, 6) = osszeg / szamlalo
End Sub
```

Faktoriális számítása

```
Sub Faktoriális()
Dim n%, i%
Dim valasz$
Dim f As Double
Do
    n = InputBox(„n=?”, „Faktor számolás”, 2)
    f = 2
    For i = 2 To n
        f = f * i
    Next i
    MsgBox „ „ & n & „!=„ & f
    valasz = InputBox(„Újabb számolás? (i/n)”, „Kérdés”, „i”)
Loop While valasz = „i”
End Sub
```

Szubrutin alkalmazása

Oszthatóság vizsgálata

```
Sub szinezo()
Dim i%, k%, n%
For i = 1 To 10
    For k = 1 To 10
        n = Int(Rnd * 100 + 1)
        Cells(i, k) = n
        If Int(n / 3) = n / 3 Then szin(„k”, i, k)
        If Int(n / 5) = n / 5 Then szin(„z”, i, k)
        If Int(n / 7) = n / 7 Then szin(„p”, i, k)
    Next k
Next i
End Sub
```

```
Sub szin(s As String, y%, x%)
If s = „z” Then Cells(y, x).Interior.Color = RGB(0, 200, 0)
If s = „k” Then Cells(y, x).Interior.Color = RGB(0, 0, 200)
If s = „p” Then Cells(y, x).Interior.Color = RGB(200, 0, 0)
End Sub
```

Összetett feladatok

Kockadobás feladat

```
Sub kockak()
Dim n%, db%
db = 0
n = 0
Do
    n = n + 1
    Cells(n, 1) = dobas
    Cells(n, 2) = dobas
    If Cells(n, 1) = Cells(n, 2) Then
    db = db + 1
        Cells(n, 1).Font.Color = RGB(255, 0, 0)
        Cells(n, 2).Font.Color = RGB(255, 0, 0)
    End If
Loop Until db > 5
```

```
MsgBox „A dobások száma: „ & n
End Sub
Function dobas() As Integer
    dobas = Int((Rnd * 6) + 1)
End Function
```

Kockadobás újra feladat

```
Sub kockak()
Dim n%, utolso%, i%, db%, max%, osszeg%
Cells(1, 3) = „legnagyobb dobás=„
For i = 1 To 100
    utolso = dobas
    Cells(i, 1) = utolso
    osszeg = utolso
    db = 1
    Do While utolso = 6
        utolso = dobas
        db = db + 1
        Cells(i, db) = utolso
        osszeg = osszeg + utolso
    Loop
    If max < osszeg Then max = osszeg
Next i
Cells(1, 5) = max
End Sub
```

```
Function dobas() As Integer
    dobas = Int((Rnd * 6) + 1)
End Function
```

Táblázat feladat

```
Sub tabla()
Dim i%, k%
Dim sor%, oszlop%
sor = InputBox(„sorok száma?”, „Adatbekérés”, 5)
oszlop = InputBox(„oszlopok száma?”, „Adatbekérés”, 6)
For k = 1 To oszlop
    Cells(1, k) = k + 1
    For i = 1 To sor
        Cells(i + 1, k) = veletlen(k + 1)
        If Int(Cells(i + 1, k) / 2) = Cells(i + 1, k) / 2 Then
            Cells(i + 1, k).Font.Color = RGB(250, 0, 0)
        End If
    Next i
Next k
Cells(1, 1) = „max=„
End Sub

Function veletlen(max As Integer) As Integer
    veletlen = Int((Rnd * max) + 1)
End Function
```

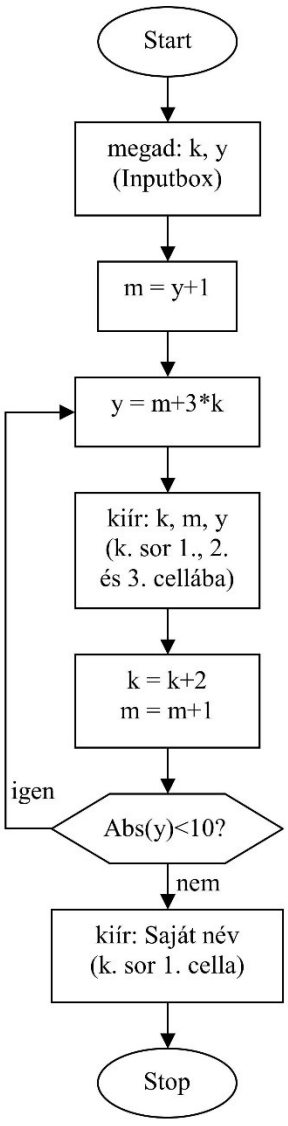
3.3 Korábbi zárthelyi feladatsorok

3.3.1 2009 tavaszi feladatsorok

3.3.1.1 1. feladatsor

1. diger – 2009. tavasz

1. Írjon VBA programot az alábbi blokkdiagram alapján! A „k” változót Integer, „y” és „m” változókat Double típusúnak definiálja! Az InputBox paranccsal történő adatbekérdezés során „k” értéke legyen 1, míg „y” -6.5! Futtassa le a programot! (6. pont)



2. Atomemissziós módszernél az intenzitás (I) és a koncentráció (c) között az $I = a \times c^b$ összefüggés áll fenn, ahol „a” és „b” konstansok. A „c (mg/dm^3)” oszlopban megadott koncentrációk esetén a mért intenzitásokat az „$I_{mért}$” oszlop tartalmazza. Az „a” és „b” konstans értékét tartalmazó M2 és M3 cellák segítségével számítsa ki, hogy a = 0.1 és b = 1 esetben mekkora intenzitást kellett volna mérni („$I_{számolt}$” oszlop). Az „$(I_m-I_{sz})^2$” oszlopban számítsa ki az adott koncentrációhoz tartozó mért és számított intenzitások különbségének a négyzetét, majd a kapott értékek összegét számítsa ki a J8 cellában. Végül a „Solvert”-t használva úgy számítson ki új „a” és „b” értékeket, hogy J8 cella értéke a lehető legkisebb legyen. A megoldás megtalálásakor a Solverrel szúrassa be „A kiszámított értékekkel” az eredményjelentést tartalmazó munkalapot. (4 pont)

	G	H	I	J	K	L	M	N
1	c (mg/dm^3)	$I_{mért}$	$I_{számolt}$	$(I_m-I_{sz})^2$				
2	0,00	0,000				a =	0,100	
3	1,00	0,090				b =	1,000	
4	2,20	0,202						
5	3,10	0,273						
6	5,00	0,440						
7								
8			Összesen =					
9								

3.3.1.2 2. feladatsor

2. diger – 2009. tavasz

1. Adott az f(k)=3/(k^2+5k+6) függvény. Írja meg a „Megoldas1” VBA programot a $\text{Sum} = \sum_{k=1}^{m} f(k) = f(1) + f(2) + ... + f(m)$ összeg kiszámítására. A programban InputBox-szal olvastassa be m és p értékét, mely utóbbi az a legkisebb k érték, amelytől kezdődően a program az összeg (Sum) értékét is kiíratja.

   A programban használjon For-Next ciklusutasítást, m, p és k változókat Integer-nek, Sum-ot Double-nak definiálja. Az f(k) értékét „Function”-nal számolja ki (függvény neve: f, single típusú)! A program kiíratása a „Munka1” munkalapon történjen és az itt megadott képnek (mely m=5 és p=3 esetét mutatja) megfelelő szerkezetű eredményt adja! (6 pont)

	A	B	C
1	k	f(k)	Sum
2	1	0,25	
3	2	0,15	
4	3	0,1	0,5
5	4	0,07143	0,57143
6	5	0,05357	0,625
7			
8	p =	3	
9	m =	5	

2. Hozzon létre új általános modult, majd abban írja át az 1. feladatban létrehozott programot úgy („Megoldas2”), hogy a For-Next ciklus helyett Do While-Loop ciklust, az f „Function” helyett pedig f „Sub”-ot alkalmazzon! Az f „Sub” által kiszámított értéket a „Megoldas2” programban az fki változó (single típusú) tárolja! A programot a „Munka2” munkalapon futtassa. (3 pont)
3. A feladatot a For-Next, vagy a Do-Loop ciklus alkalmazásával oldaná meg akkor, ha a program nem a k = m határig, hanem addig futna, amíg a Sum <= 0.6 feltétel teljesül? Indokolja meg válaszát! (1 pont)

3.3.1.3 3. feladatsor

3. diger – 2009. tavasz

1. Írja meg a „Haromszog” nevű VBA programot, amely kiszámolja a térben elhelyezkedő három pontot (A, B, C) összekötő AB, AC és BC vektorok ismeretében a háromszög egyes oldalainak hosszúságát. A program először For-Next ciklus alkalmazásával olvassa be az „AB.txt” és „AC.txt” file-ból az AB és AC vektorok 3-3 adatát az „intAB” és „intAC” (integer típusú) tömbökbe. A program utána számolja ki a BC vektor („intBC”, integer) paramétereit, amelyek AC és AB vektorok megfelelő paramétereinek a különbségei. Majd a program a „Hossz” function (single típusú, bemenő paraméterek a megfelelő tömbök adatai) alkalmazásával számolja ki az egyes oldalak hosszúságát (sngAB, sngAC és sngBC, mind single típusú) az alábbi összefüggés segítségével:

$$\text{Hosszúság} = \sqrt{(x^2 + y^2 + z^2}\,,$$

ahol x, y és z az adott vektor x-, y- és z-irányú komponense. Végezetül a számított értékeket írassa ki az Oldalhossz.txt file-ba az alábbi módon: sngAB = érték, sngAC = érték, sngBC = érték. (4.5 pont)

2. Írja meg a „Dinamikus” nevű VBA programot, amely először a dinamikus intTomb tömbbe (integer típusú) beolvassa a „Szamok.txt” file-ban lévő „n” darab értéket („n” a file-ból beolvasott adatok száma), majd kiszámítja a long típusú lngTomb „n” darab értékét az alábbi összefüggés alapján:

$$\ln gTomb(i) = \text{int}\,Tomb(i) \times k\,,$$

ahol k = n+1-i. (Példa: 20 elemű file esetén, ha a intTomb(1) = 5, intTomb(20) = 6, akkor lngTomb(1) = 5 × 20 = 100 és intTomb(20) = 6 × 1 = 6.) Utána számítsa ki „lngOsszeg” értékét a

$$\ln gOsszeg = \sum_{i=1}^{n} \ln gTomb(i)$$

egyenlet alapján, végezetül a MsgBox utasítással írassa ki „lngOsszeg” értékét. (5.5 pont)

3.3.1.4 4. feladatsor

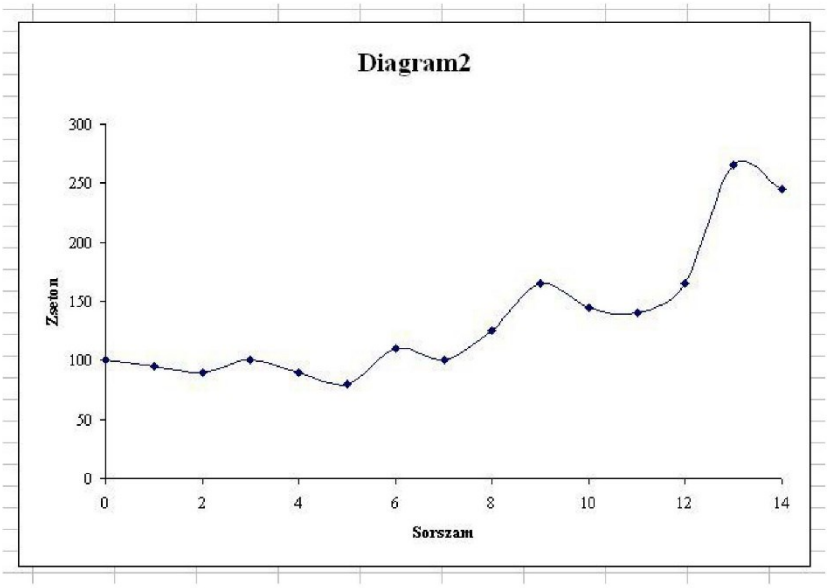
4. diger (pótdiger) – 2009. tavasz

1. Írjon a „Module 1” modulba VBA programot, amely megnyitja a „Poker-be.txt” file-t, először beolvassa kiindulási zsetonösszeget az integer típusú „intKiindulasi” paraméterbe, majd az „intSorszam()”, „intBefizetes()” és az „intNyeremeny()” dinamikus tömbökbe (integer típusúak) beolvassa a játék sorszámát (első oszlop), az adott játék során befizetett zseton mennyiségét (második oszlop), valamint a nyereményt (harmadik oszlop). A long típusú „lngOsszPenz()” dinamikus tömbbe számítsa ki az adott játék után rendelkezésre álló zsetonok összegét az alábbi képlet alapján:

$$lngOsszPenz(intI) = lngOsszPenz(intI-1) - intBefizetes(intI) + intNyeremeny(intI)$$

(intI = 1 esetén „intKiindulasi” paraméterből vonja ki/adja hozzá az első játék során befizetett/nyert zseton összegét). Ezután az „A1” cellába írja be a „Sorszám”, a „B1” cellába a „Zseton” sztringet, az „A2” cellába nulla, a „B2” cellába „intKiindulasi” értékét, majd az „A” és „B” oszlopok következő celláiba az „intSorszam()”, illetve „lngOsszPenz()” tömböki megfelelő értékeit. Utána nyissa meg írásra a „Poker-ki.txt” file-t, melybe írja bele a „Jatekszam = ”, a „Nyeremeny = ” és az „Egy jatekra eso nyeremeny = ” sztringeket, valamint a játékok számát, a nyereményt (játék végén rendelkezésre álló zsetonösszeg és „intKiindulasi” különbsége), valamint a kettő hányadosát.
Utána a program az InputBox paranccsal kérdezze meg, hogy a felhasználó meg akarja-e jeleníteni a Sorszám/Zseton adatokat. Amennyiben „i” (igen) a válasz, a program hívja meg a „Module 1”-ben lévő Diagram2() eljárást, amelyet a „Module 2”-ben lévő Diagram() eljárás „Module 1”-be történő átmásolásával, majd a megfelelő paraméterek átírásával hozzon létre. A Diagram2() eljárás bemenő paramétere a legnagyobb játék sorszáma legyen.

	A	B
1	Sorszam	Zseton
2	0	100
3	1	95
4	2	90
5	3	100
6	4	90
7	5	80
8	6	110
9	7	100
10	8	125
11	9	165
12	10	145
13	11	140
14	12	165
15	13	265
16	14	245

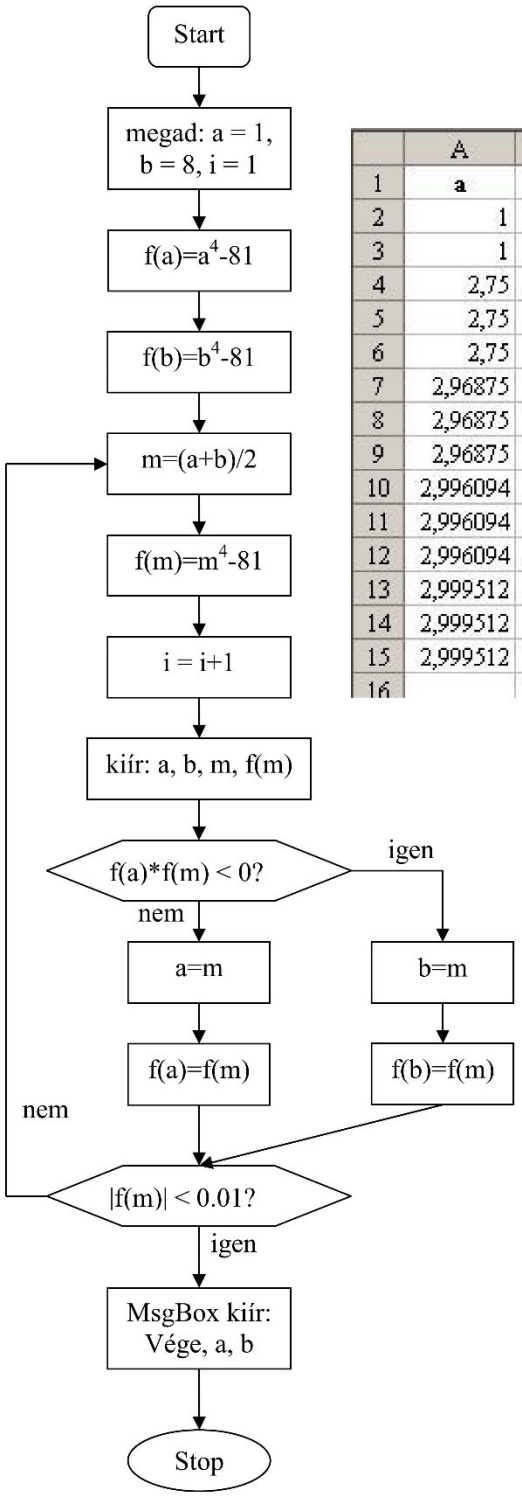


3.3.2 2009 őszi feladatsorok

3.3.2.1 1. feladatsor, A csoport

Név: **Tankör:** **Diger-1, A csoport – 2009 ősz**

1. Az alább megadott blokkdiagram alapján készítse el a „Feladat_1a” programot a Do - Loop While, vagy Do - Loop Until ciklus utasítás használatával az „$x^4 - 81 = 0$” egyenlet gyökének intervallum-felező módszerrel való meghatározására. Az „a” és „b” változók kezdőértékét InputBox segítségével adja meg. A program az „a”, „b”, „m” és „fm” értékét az alábbi ábrán látható módon írja ki (a táblázat fejléce nem programozandó). Utána írja át a program megfelelő részét úgy, hogy a a Do - Loop ciklusutasítás helyett If-Then és GoTo utasításokat alkalmazzon. A program megírása során az alábbi változókat alkalmazza: i (integer), a, b, m (mind single), fa, fb, fm (mind double). (13+2 pont).

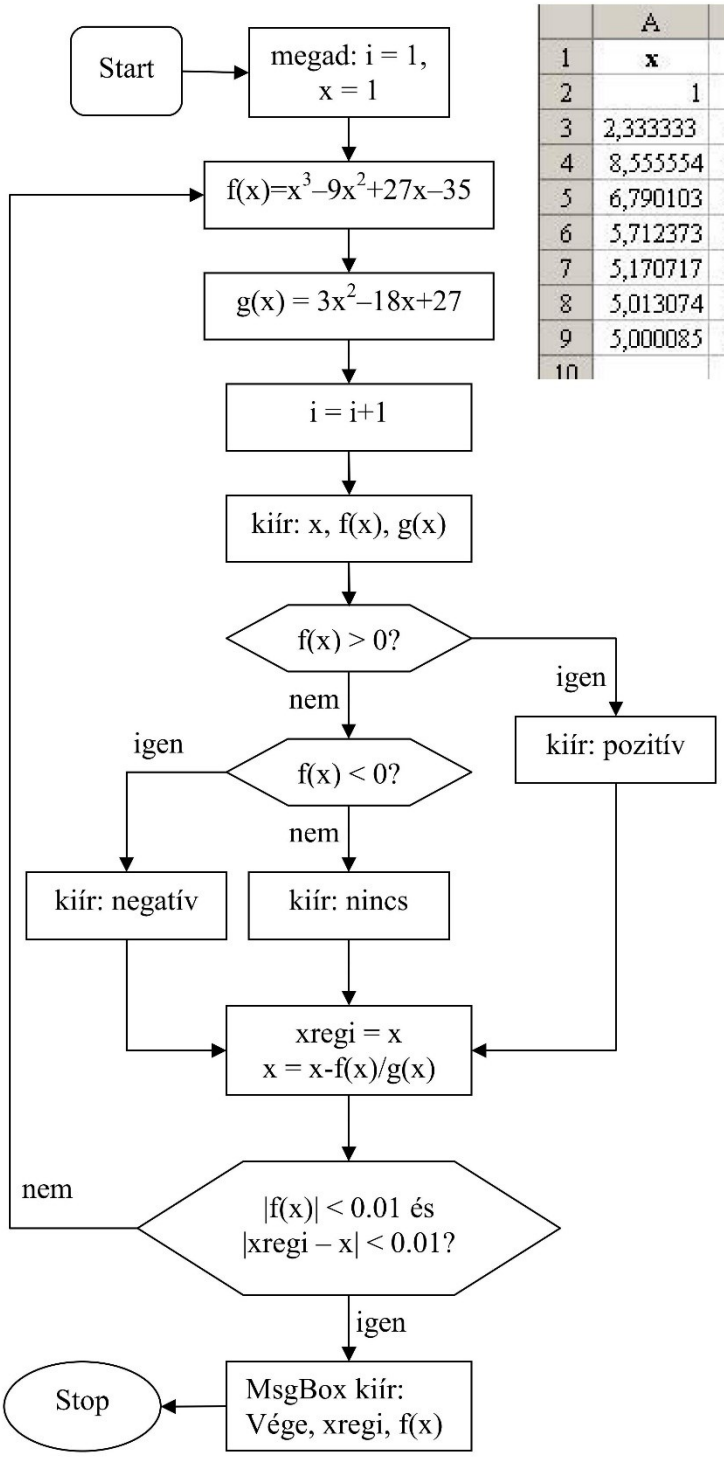


	A	B	C	D
1	**a**	**b**	**m**	**f(m)**
2	1	8	4,5	329,0625
3	1	4,5	2,75	-23,8086
4	2,75	4,5	3,625	91,67603
5	2,75	3,625	3,1875	22,22878
6	2,75	3,1875	2,96875	-3,32263
7	2,96875	3,1875	3,078125	8,772849
8	2,96875	3,078125	3,023438	2,561068
9	2,96875	3,023438	2,996094	-0,42105
10	2,996094	3,023438	3,009766	1,059849
11	2,996094	3,009766	3,00293	0,31687
12	2,996094	3,00293	2,999512	-0,05272
13	2,999512	3,00293	3,001221	0,131916
14	2,999512	3,001221	3,000366	0,039558
15	2,999512	3,000366	2,999939	-0,00659
16				

3.3.2.2 1. feladatsor, B csoport

Név: **Tankör:** **Diger-1, B csoport – 2009 ősz**

1. Az alább megadott blokkdiagram alapján készítse el a „Feladat_1b” programot a Do - Loop While, vagy Do - Loop Until ciklus utasítás használatával az „$\mathbf{x^3 - 9\,x^2 + 27\,x - 35 = 0}$” egyenlet gyökének Newton módszerrel való meghatározására. Az „x” változó kezdőértékét InputBox segítségével adja meg. A program az „x”, „f(x)”, „g(x)” és a „hiba típusa” értékét az Excel cellákba az alábbi ábrán látható módon írja ki (a táblázat fejléce nem programozandó).
   Utána írja át a program megfelelő részét úgy, hogy a a Do - Loop ciklusutasítás helyett If-Then és GoTo utasításokat alkalmazzon. A program megírása során az alábbi változókat alkalmazza: i (integer), x, xregi (single), fx, gx (mind double). (13+2 pont).



	A	B	C	D
1	**x**	**f(x)**	**g(x)**	**Hiba**
2	1	-16	12	negatív
3	2,333333	-8,2963	1,333334	negatív
4	8,555554	163,4677	92,59255	pozitív
5	6,790103	46,44438	43,09464	pozitív
6	5,712373	11,95485	22,07091	pozitív
7	5,170717	2,228442	14,13603	pozitív
8	5,013074	0,157915	12,1574	pozitív
9	5,000085	0,001019	12,00102	pozitív
10				

3.3.2.3 2. feladatsor, A csoport

Név: **Tankör:** **Diger-2, A csoport – 2009 ősz**

1. Írja meg a „Haromszog” nevű VBA programot, amely először For-Next ciklus alkalmazásával olvassa be a „Pontok.txt” file-ból az A, B és C háromdimenziós pontok 3-3 koordinátáját (a pontok adatai a file soraiban találhatóak) az „Adatok” tömbbe (integer, méret = 3×3). Utána kiszámolja a pontokat összekötő vektorok koordinátáit az „AB”, „AC” és „BC” tömbökbe (mind integer, méret = 3), végül a „Hossz” nevű single típusú Function (3 paramétere a vektor 3 integer típusú koordinátája, és neve a vektor hosszának értékét veszi fel) segítségével számolja ki az egyes oldalak hosszúságát („ABhossz”, „AChossz” és „BChossz”, mind single típusú) az alábbi összefüggés segítségével:

$$\text{Hosszúság} = \sqrt{(x^2 + y^2 + z^2)},$$

ahol x, y és z az adott vektor három koordinátája. Végezetül a számított értékeket írassa ki az alábbi módok egyikén:
a) vagy az „Oldalhossz.txt” file-ba az alábbi módon: ABhossz = érték, AChossz = érték, BChossz = érték
b) vagy a munkalap celláiba az alábbi ábrán látható módon. Ebben az esetben a program keresse meg a leghosszabb oldalt és azt jelezze a „C” oszlop megfelelő cellájában. (12+3 pont)

Pontok.txt tartalma:
0, 0, 0 ← A pont
2, 4, 5 ← B pont
6, 4, 1 ← C pont

Oldalhossz.txt tartalma:
"ABhossz = ",6.708204
"AChossz = ",7.28011
"BChossz = ",5.656854

	A	B	C
1	ABhossz =	6,7082	
2	AChossz =	7,2801	leghosszabb
3	BChossz =	5,6569	

2. feladatsor, B csoport

Név: **Tankör:** **Diger-2, B csoport – 2009 ősz**

1. Írja meg a „Munkaber” nevű VBA programot, amely először For-Next ciklus alkalmazásával olvassa be az „Dolgozok.txt” file-ból a „Nev” tömbbe (string, méret = 5) a dolgozók nevét, a „Munka” tömbbe (long, méret = 5×3) a dolgozók órabérét, a ledolgozott órák számát és a jutalmat. Utána kiszámolja az egyes személyeknek kifizetendő bruttó összeget a „Kifizetes” tömbbe (long, méret = 5) a „Szamol” nevű long típusú Function (3 paramétere az adott munkás órabére, a ledolgozott órák száma és a jutalom) segítségével:

Kifizetés = órabér × ledolgozott órák száma + jutalom,

valamint az „Osszesen” (long) változóba kiszámolja az összesen kifizetendő bruttó munkabért.
Végezetül dolgozók nevét és a számított bruttó munkabér összegét írassa ki az alábbi módok egyikén:
a) vagy az „Osszegzes.txt” file-ba az alábbi módon: Név = kifizetendő összeg,
b) vagy a munkalap celláiba az alábbi ábrán látható módon. Ebben az esetben figyeljen rá, hogy a „B” oszlopban a fizetendő összegek ezredrésze szerepel. (12+3 pont)

Dolgozok.txt tartalma:
Pisti, 1200, 160, 8000
Jani, 1500, 120, 6000
Emese, 1300, 160, 10000
Robi, 900, 180, 15000
Ancsa, 1000, 140, 7000

Osszegzes.txt tartalma:
"Pisti = ",200000
"Jani = ",186000
"Emese = ",218000
"Robi = ",177000
"Ancsa = ",147000
"Osszesen = ",928000

	A	B	C
1	Pisti	200	eFt
2	Jani	186	eFt
3	Emese	218	eFt
4	Robi	177	eFt
5	Ancsa	147	eFt
6	Osszesen	928	eFt

3.3.2.4 3. feladatsor, A csoport

Név: **Tankör:** **Diger-3, A csoport – 2009 ősz**

1. Egészítse ki az alábbi programrészletet a jobb oldali oszlopban lévő 1-16. számú részletekkel úgy, hogy a program futtatása után eredményként a baloldali táblázatot kapjuk. (6 pont)

	A	B
1	2	1,414214
2	4	16
3	6	2,44949
4	8	64
5	10	3,162278
6	12	144
7	14	3,741657
8	16	256
9	18	4,242641
10	20	400
11	Összesen =	895,0103

```
Sub Feladat3a1()

Dim ___ Integer
Dim ___ Integer
Dim ___ Single

___ = 0

For i = 2 ___ 20 ___ 2
___ = i / 2
___ = i
___
If i ___ 4 = 0 Then
   Cells(j, 2) = ___
   ___
   Cells(j, 2) = ___(i)
___
___ = a + Cells(j, 2)
___ i

Cells(11, 1) = "Összesen ="
___ = a

End Sub
```

1) a
2) a
3) j
4) a As
5) i As
6) j As
7) To
8) Mod
9) Step
10) Else
11) Sqr
12) Cells(j, 1)
13) Cells(11, 2)
14) End If
15) Next
16) i ^ 2

2. Írja meg a Feladat3a2 nevű programot, amely az „Adat3a2.txt” file-ban lévő ismeretlen számú adatot (ciklusban, egyenként) beolvassa a „Be” nevű változóba (integer), majd amennyiben a „Be” érékének négyzetgyöke is egész szám, akkor „Be” értékét beírja a „Tomb” nevű dinamikus tömbbe (integer). A Tomb(0) helyen tárolja azt a számot, amely megmutatja, hogy az „Adat3a2.txt” file-ból mennyi számot írtunk át a dinamikus tömbbe. Végül a program a „Tomb” tartalmát a jobboldali táblázatban látható módon írja ki („A1” és „A2” cella tartalma is programozandó). A feladat megoldása során használja a kerekítésre szolgáló „Round” belső függvényt (pl: Round(5.6) értéke 6 lesz). (9 pont)
Amennyiben a dinamikus tömb helyett fix méretű tömböt használ, legfeljebb 6 pontot kaphat a feladatra.

	A	B
1	Adatszam =	8
2	Szamok =	9
3		81
4		169
5		196
6		64
7		36
8		144
9		121

Az „Adat3a2.txt” file tartalma (vastagon és dőlten szedve a kiválasztott számok):
5, ***9***, 7, 132, 199, 31, ***81***, ***169***, 45, ***196***, 62, ***64***, 108, 86, 39, 116, 55, ***36***, ***144***, 18, ***121***, 39, 164, 183, 24, 78

3.3.2.5 3. feladatsor, B csoport

Név: **Tankör:** **Diger-3, B csoport – 2009 ősz**

1. Egészítse ki az alábbi programrészletet a jobb oldali oszlopban lévő 1-16. számú részletekkel úgy, hogy a program futtatása után eredményként az alábbi táblázatot kapjuk. (6 pont)

	A	B	C	D	E	F	G	H	I	J	K
1	3	6	9	12	15	18	21	24	27	30	Összeg =
2	1,732051	3,320117	3	11,02318	3,872983	36,59823	4,582576	121,5104	5,196152	403,4288	594,2645

```
Sub Feladat3b1()

Dim ___ Integer
Dim ___ Integer
Dim ___ Single

___ = 0

For i = 3 ___ 30 ___ 3
___ = i / 3
___ = i
If i ___ 6 = 0 Then
   Cells(2, j) = ___(i / 5)
   ___
   Cells(2, j) = ___
___
___ = b + Cells(2, j)
___ i

Cells(1, 11) = "Összeg ="
___ = b

End Sub
```

1) b
2) b
3) j
4) b As
5) i As
6) j As
7) To
8) Mod
9) Step
10) Else
11) Exp
12) Cells(1, j)
13) Cells(2, 11)
14) End If
15) Next
16) i ^ 0.5

2. Írja meg a Feladat3b2 nevű programot, amely egy személy bevételeit és kiadásait tartalmazó „Adat3b2.txt” file-ban lévő ismeretlen számú adatot (ciklusban, egyenként) beolvassa a „Be” nevű változóba (long), majd amennyiben a „Be” abszolutértéke nagyobb, mint 10000, akkor „Be” értékét beírja a „Sok” nevű dinamikus tömbbe (long). A Sok(0) helyen tárolja azt a számot, amely megmutatja, hogy az „Adat3b2.txt” file-ból mennyi számot írtunk át a dinamikus tömbbe. Végül a program a „Sok” tömb tartalmát a jobboldali táblázatban látható módon írja ki („A1” és „A2” cella tartalma is programozandó). (9 pont)

	A	B
1	Tetelszam =	4
2	Tetelek =	152396
3		-12000
4		25430
5		-20000

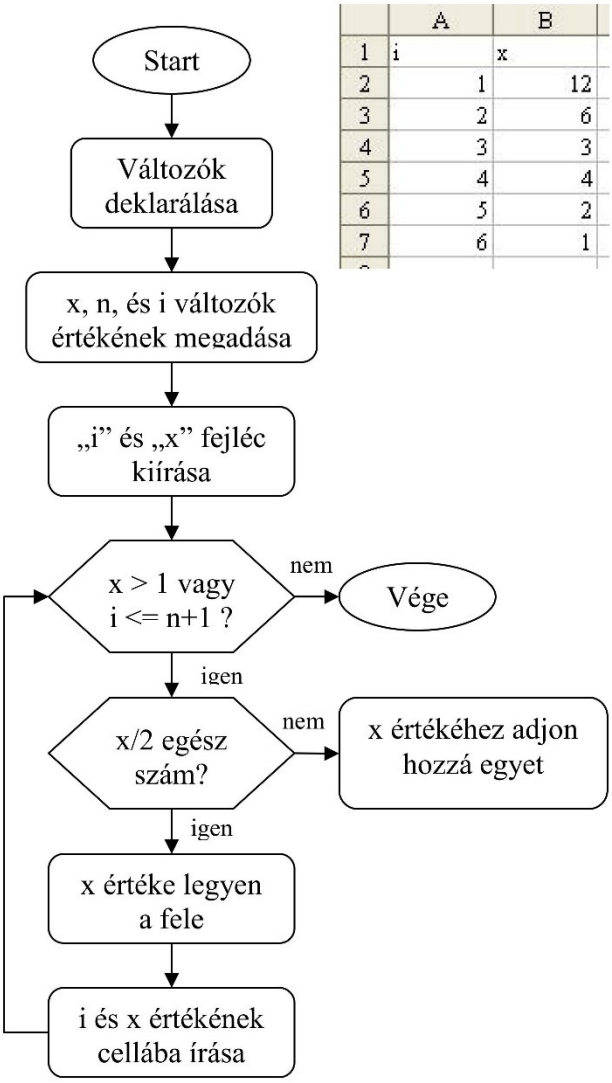
Amennyiben a dinamikus tömb helyett fix méretű tömböt használ, legfeljebb 6 pontot kaphat a feladatra.

Az „Adat3b2.txt” file tartalma (vastagon és dőlten szedve a kiválasztott számok):
152396, ***-12000***, -6458, -3420, ***25430***, ***-20000***, 7389, -9500, -25, -679, -3100, -25, -675, 3960, 2180, -5267

3.3.2.6 4. feladatsor

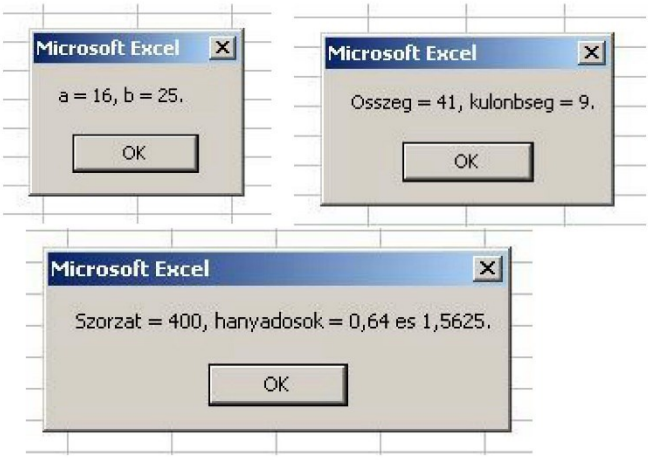
Név: **Tankör:** **Diger-4 – 2009 ősz**

1. Írja meg a Feladat1 programot a mellékelt blokkdiagram alapján úgy, hogy a program futása után eredményként (x = 11, n = 5 esetben) a jobboldali ábrát kapjuk. A program írása során az „x”, „n”, és „i” integer típusú változókat használja. A program „x” és „n” értékét InputBox paranccsal kérje be, míg „i” kezdőértéke egy legyen. A program írása során Do-While Loop ciklust alkalmazzon. (9 pont)

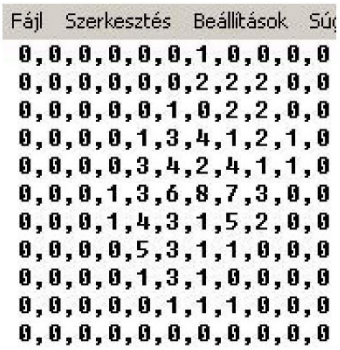


	A	B
1	i	x
2	1	12
3	2	6
4	3	3
5	4	4
6	5	2
7	6	1

2. Írja meg a Feladat2 nevű programot, amely először beolvassa „a” változó (integer) értékét az „A1”, „b” változó (integer) értékét a „B1” cellából, majd az „Osszeadas”, „Szorzas” és „Kivonas” függvények, valamint az „Osztas” szubrutin segítségével kiszámolja az a+b („Osszeg” változó, long) a×b („Szorzat” változó, long), |a-b| („Kulonbseg” változó, long), valamint a/b és b/a („Hanyados1” és („Hanyados2” változók, single) értékét. Végezetül a MsgBox parancs segítségével a változók értékeit alábbi ábrán látható módon írassa ki. (10 pont)



3. Írja meg a Feladat3 nevű programot, amely először a 11×11 méretű „Tomb” tömböt feltölti 0 értékekkel, majd „x” és „y” változóknak 6 értéket adja és a Tomb(x,y) értékét 1-re állítja. Utána 100 ciklus mindenegyes ciklusában „x” és „y” értékét -1, 0, vagy 1 értékkel változtassa meg (a -1, 0 és 1 értékeket az „Int(3 * Rnd) - 1” kifejezéssel állíthatja elő). Amennyiben „x” vagy „y” értéke kisebb lesz, mint 1, akkor a változó értékéhez adjon hozzá 11-t, míg ha „x” vagy „y” értéke nagyobb lesz, mint 11, akkor a változó értékéből vonjon ki 11-t, majd a Tomb(x,y) értékét növelje meg 1-vel. A 100 ciklus után a „Tomb” tartalmát a jobboldali ábrán látható módon írja ki az Eredmeny.txt file-ba. (11 pont)



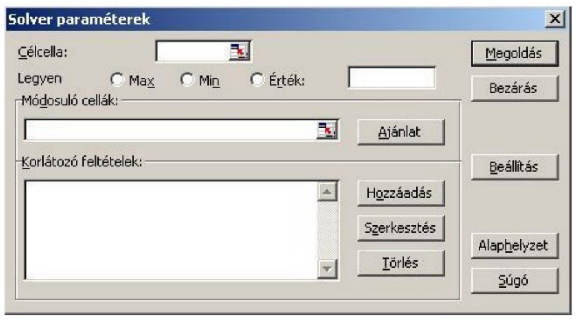
3.3.3 2010 őszi feladatsorok

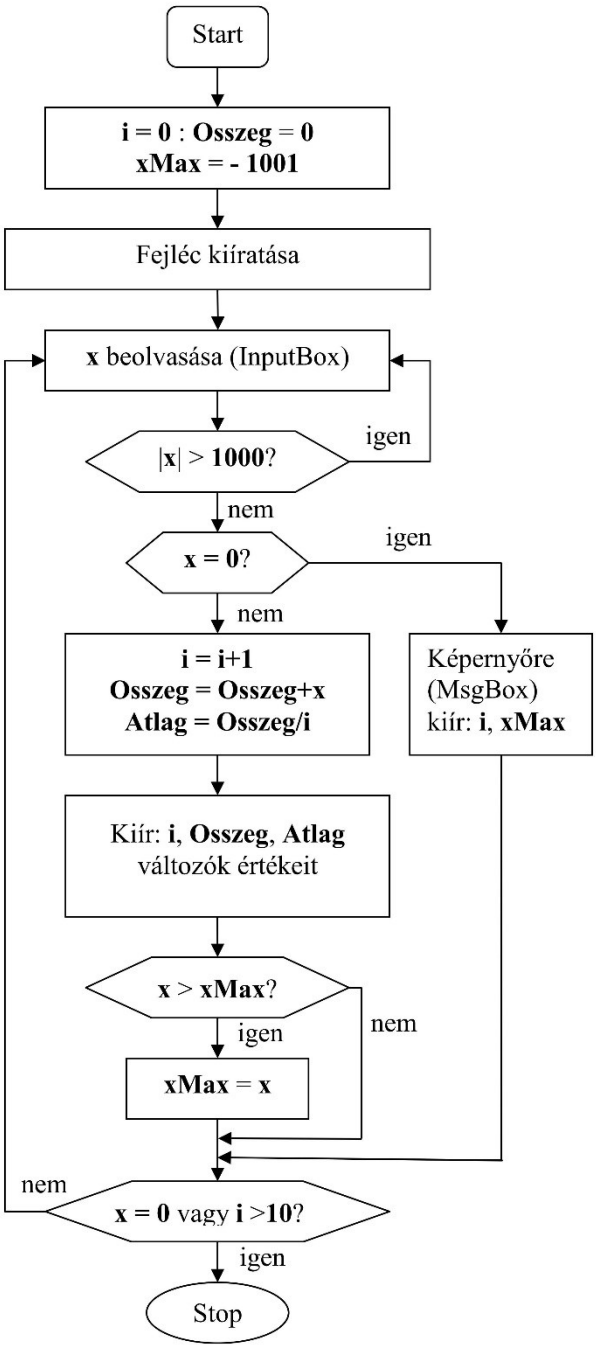
3.3.3.1 1. feladatsor, A csoport

Név: **Tankör:** **Diger-1, A csoport – 2010 ősz**

1. Egy téglatest élei: „a”, „b” és „c”. Az élek hosszáról tudjuk, hogy a + b + c = 12 egység és b = a / 2 + 2. A Solver segítségével keressük azt a téglatestet, amelynek a térfogata a legnagyobb. Adja meg, hogy milyen képletet kell írnunk a B3, B4 és B5 cellákba, melyik legyen a célcella, mire állítsuk be a „Legyen” értékét. (lásd a mellékelt ábrák) (2 pont)

	A	B
1	"a" él=	2
2	"b" él=	3
3	"c" él=	7
4	Térfogat=	42





2. A baloldalt megadott blokkdiagram alapján készítse el a „Beolvasas” programot a Do - Loop While ciklus utasítás használatával. Az „**x**” változó kezdőértékét InputBox segítségével adja meg. A program a táblázat fejlécét, valamint az „**i**”, „**Osszeg**” (a beadott „**x**” értékek összege) és „**Atlag**” (a beadott „**x**” értékek átlaga, **Osszeg** / **i**) értékét az alábbi ábrán látható módon Excel cellákba írja ki, míg „**i**” és „**xMax**” értékét a MsgBox paranccsal a képernyőre írja ki. A program megírása során használt változók közül **i, x** és **xMax** integer, **Osszeg** long és **Atlag** single legyen. Milyen **xMax** értéket ír ki a program, ha a beolvasott x értékek (az alábbi ábrán megadott kiiratásnak megfelelően) rendre 5, 9, -3, 42, -35, 6 és 0? (13 pont).

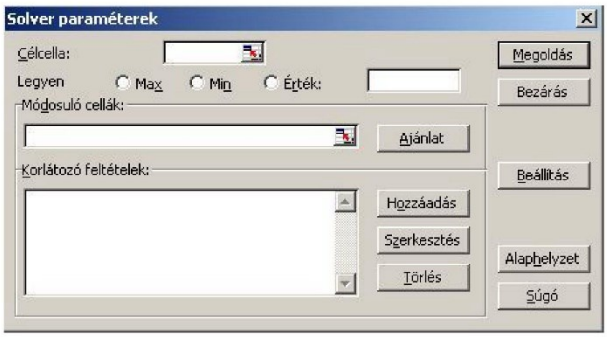
	A	B	C
9			
10	i	Osszeg	Atlag
11	1	5	5
12	2	14	7
13	3	11	3,66667
14	4	53	13,25
15	5	18	3,6
16	6	24	4

3.3.3.2 1. feladatsor, B csoport

Név: **Tankör:** **Diger-1, B csoport – 2010 ősz**

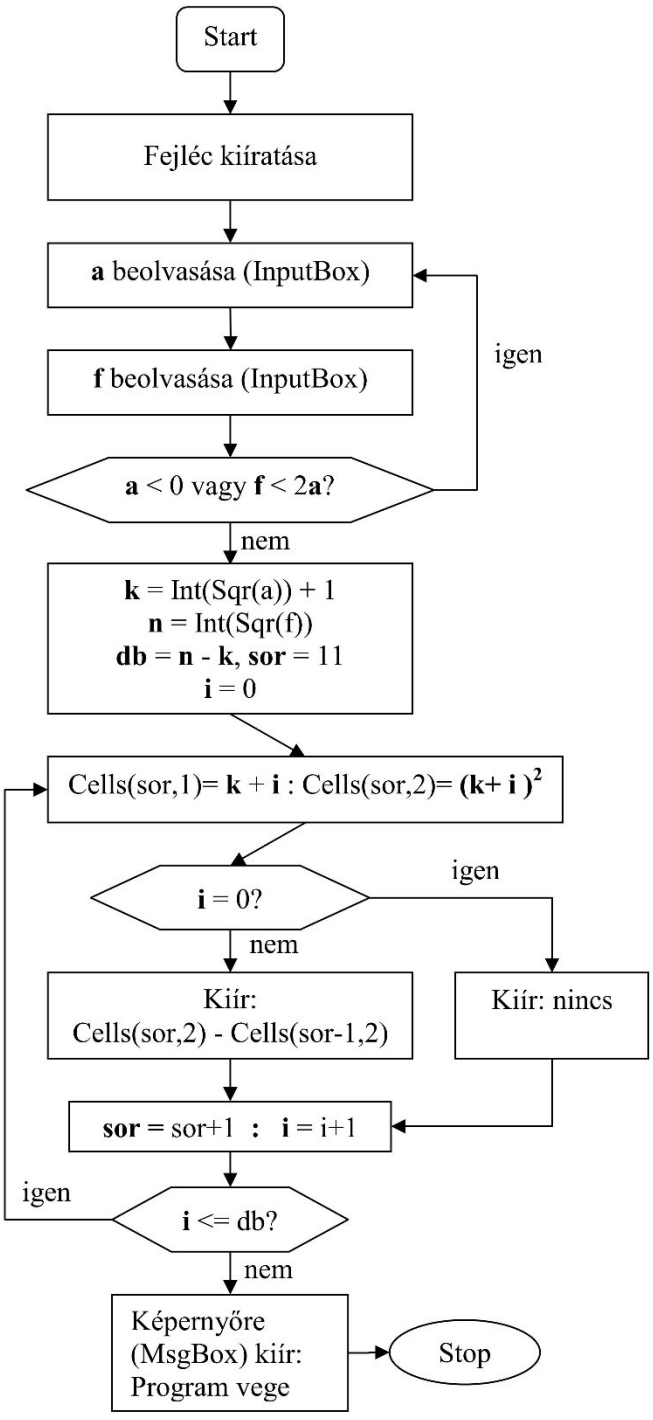
1. Az $y = a * x^2 + b * x + c$ másodfokú egyenlet „a”, „b” és „c” együtthatóiról tudjuk, hogy a + b + c = 9 és a=b/4. A Solver segítségével azokat az együtthatókat keressük, melyeknél az egyenlet $D=b^2-4ac$ diszkriminánsa zérus. Adja meg, hogy milyen képletet kell írnunk a B2, B4 és B5 cellákba, melyik legyen a célcella, mire állítottuk be a „Legyen” értékét. (lásd a mellékelt ábrák) (2 pont)

	A	B
1		
2	a=	0,5
3	b=	2
4	c=	6,5
5	D=	-9



2. Az jobboldalt megadott blokkdiagram alapján készítse el a „Negyzetek” programot a Do - Loop While ciklus utasítás használatával. Az „**a**” és „**f**” változók kezdőértékét InputBox segítségével adja meg. Utána a program a táblázat fejlécét, valamint a „**k** + **i**”, „$(\mathbf{k} + \mathbf{i})^2$” és „$(\mathbf{k} + \mathbf{i})^2 - (\mathbf{k} + \mathbf{i} - 1)^2$” értékeket az alábbi ábrán látható módon (**a** = 300, **f** = 600) Excel cellákba írja ki, míg a „Program vege” üzenetet a MsgBox paranccsal a képernyőre írja ki. Az Int belső függvény paraméterének egész részét számolja, pl. Int(5.6) értéke 5. A program megírása során használt változók mind integer típusúak legyenek. (13 pont).

	A	B	C
9			
10	Szam	Negyzet	Kulonbseg
11	18	324	nincs
12	19	361	37
13	20	400	39
14	21	441	41
15	22	484	43
16	23	529	45
17	24	576	47



3.3.3.3 2. feladatsor, A csoport

Név: **Tankör:** **Diger-2, A csoport – 2010 ősz**

1. Írja meg a **Megoldas1** nevű VBA programot, az adott f(k)=3/(Ak^2+Bk+C) függvény aktuális értékeinek, és a $\text{Sum} = \sum_{k=1}^{m} f(k) = f(1) + f(2) + ... + f(m)$ összeg kiszámítására.

A másodfokú polinom A, B, C együtthatóit rendre a jobb oldalon látható *egyutthatok.txt* tartalmazza. Az együtthatókat az ADAT nevű kétindexes (5 x 3) méretű tömbbe olvastassa be, és írja ki az Excel cellákba. A függvény értékek és az egyes függvényekhez tartozó SUM értékek tárolására ERTEK kétindexes dinamikus (5 x m), ill. SUM egyindexes (5) tömböket deklaráljon. Az **m** értékét Inputbox-szal olvastassa be (legyen 4).
A függvényértékek számításához deklaráljon F nevű, single értékű Function-t, melynek 4 paramétere, k, A, B és C Integer típusúak.
A SUM tömb elemeit írja ki *osszegek.txt* fájlba.
Az Ertek tömb elemeit írassa ki az Excel munkalapra az alábbi, *m=4* értéknek megfelelő ábrán látható módon. **(Összesen: 15 pont)**

Együtthatok.txt

A	B	C
1	5	6
2	3	4
3	3	-2
2	4	-1
4	4	4

	A	B	C	D	E	F	G	H	I
1	**A**	**B**	**C**		**k =**	**1**	**2**	**3**	**4**
2	1	5	6			0,25	0,15	0,10	0,07
3	2	3	4			0,33	0,17	0,10	0,06
4	3	3	-2			0,75	0,19	0,09	0,05
5	2	4	-1			0,60	0,20	0,10	0,06
6	4	4	4			0,25	0,11	0,06	0,04

2. feladatsor, B csoport

Név: **Tankör:** **Diger-2, B csoport – 2010 ősz**

Fájl Szerke
0, 0
3, 4
8, 7
12, 5

1. Írja meg a **D2_B** nevű VBA programot, amely először a *Koordinatak.txt* file-ban lévő egész számokat (lásd baloldali ábra, az egyes sorokban levő számpárok rendre egy síkbeli pont (x,y) koordinátái), beolvassa az **X** ill. **Y** dinamikus tömbökbe, a **db** változóban tárolja a beolvasott koordináta-párok darabszámát, majd definiálja a kétindexes **db** × **db** méretű **Matrix** nevű Double típusú dinamikus tömböt. A **Matrix** tömb elemeinek az értékét az alábbi módon számítsa ki:

$$Matrix(i,j) = \sqrt{(X(i) - X(j))^2 + (Y(i) - Y(j))^2}$$

Minden egyes **Matrix(i,j)** tömbelem értékét írja ki egy Excel munkalapra, az (i, j) indexeknek (sor, oszlop) szerint megfelelő cellába. A **Matrix** tömb elemeit írassa ki a *kimenet.txt* fájlba is.

Változtassa meg a **D2_B** programot úgy, hogy a **Matrix(i,j)** tömbelemek értékét a **Tavolsag** Function hívásával számítsa. Készítsen **Tavolsag** nevű, Double típusú Function-t, melynek négy formális paramétere (**a1, a2, b1, b2)** legyenek Integer típusúak (Nem szükséges az egész **D2_B** programot újra leírnia, csak a változtatandó részt.). **(Összesen: 15 pont)**

	A	B	C	D	E	F	G	H
1	X()	Y()		Számolás az alap Subrutinban				
2	0	0		0,00	5,00	10,63	13,00	
3	3	4		5,00	0,00	5,83	9,06	
4	8	7		10,63	5,83	0,00	4,47	
5	12	5		13,00	9,06	4,47	0,00	
6								
7								

3.3.3.4 3. feladatsor, A csoport

Név: **Tankör:** **Diger-3, A csoport – 2010 ősz**

1. Egészítse ki az alábbi programrészletet a jobb oldali oszlopban lévő 1-16. számú részletekkel úgy, hogy a program futtatása után eredményként a jobboldali táblázatot kapjuk. (4 pont)

```
Sub Program_D3_1A()

Dim ___
Dim j As Integer
Dim ___ Integer
Dim ___ Single

___ i = 1 ___ 15
Cells(___) = i
Cells(1, i + 1) = i
For j = i To 15
    k = ___
    Cells(___) = k
    m = Sqr(___)
    ___ m = Int(m) Then
        Cells(___) = m
        ___
        Cells(i + 1, j + 1) = ___
    ___
Next ___
Next ___

End Sub
```

1) k
2) For
3) j
4) i
5) i%
6) "-"
7) m As
8) k As
9) i ^ 2 + j ^ 2
10) If
11) To
12) End If
13) Else
14) i + 1, j + 1
15) i + 1, 1
16) j + 1, i + 1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	2	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	3	10	13	-	5	-	-	-	-	-	-	-	-	-	-	-
5	4	17	20	25	-	-	-	-	-	-	-	-	-	-	-	-
6	5	26	29	34	41	-	-	-	-	-	-	-	13	-	-	-
7	6	37	40	45	52	61	-	-	10	-	-	-	-	-	-	-
8	7	50	53	58	65	74	85	-	-	-	-	-	-	-	-	-
9	8	65	68	73	80	89	100	113	-	-	-	-	-	-	-	17
10	9	82	85	90	97	106	117	130	145	-	-	-	15	-	-	-
11	10	101	104	109	116	125	136	149	164	181	-	-	-	-	-	-
12	11	122	125	130	137	146	157	170	185	202	221	-	-	-	-	-
13	12	145	148	153	160	169	180	193	208	225	244	265	-	-	-	-
14	13	170	173	178	185	194	205	218	233	250	269	290	313	-	-	-
15	14	197	200	205	212	221	232	245	260	277	296	317	340	365	-	-
16	15	226	229	234	241	250	261	274	289	306	325	346	369	394	421	-

Adja meg, hogy a C3-as cellában milyen számértéket írt felül a benne jelenleg látható "-" karakter! (1 pont)

Fájl Szerkesztés

```
Andras,5
Balazs,4
Cecilia,2
Dezso,1
Elemer,3
Ferenc,3
Gabor,3
Hedvig,2
Ilona,4
Judit,1
Kalman,5
Laszlo,2
Melinda,3
Nora,1
Oszkar,3
Peter,4
Ramona,3
Sandor,3
Tibor,2
Ubul,3
Xaver,4
Yvonne,4
Vajk,4
Zoltan,2
```

	A	B	C	D	E
20				Jegy	Db
21	Andra	5		1	3
22	Balaz	4		2	5
23	Cecil	2		3	8
24	Dezs	1		4	6
25	Elem	3		5	2
26	Ferer	3			
27	Gabo	3			
28	Hedvi	2			
29	Ilona	4			
30	Judit	1			
31	Kalm	5			
32	Laszl	2			
33	Melin	3			
34	Nora	1			
35	Oszk	3			
36	Peter	4			
37	Ramo	3			
38	Sand	3			
39	Tibor	2			
40	Ubul	3			
41	Xaver	4			
42	Yvon	4			
43	Vajk	4			
44	Zolta	2			

2. Írja meg a **D3_2A** nevű programot, amely:
- a baloldali, bekeretezett ábrán látható „Adatok.txt” fájlban lévő ismeretlen számú név-jegy adatpárt (ciklusban, egyenként) beolvassa a **nev** nevű (String típusú) és **jegy** nevű (Integer típusú) változókba,
- a fejlécet, valamint a **nev** és **jegy** változók értékét az ábrán látható módon kiírja egy Excel munkalap celláiba,
- beolvasás közben a **Jegyek** nevű (Integer típusú), egyindexes tömbben tárolja a különböző osztályzatok darabszámát, amelyeket végül az ábrán látható módon kiír, és MsgBox használatával kiírja a jegyek átlagát is.

A program minden változóját deklarálja (típusuk megadásával)!

(10 pont)

3.3.4 2011 őszi feladatsorok

3.3.4.1 1. feladatsor, A csoport

Név: **Tankör:** **Diger-1, A csoport – 2011 ősz**

1. A megadott blokkdiagram alapján készítse el a „A1a” programot az if - then és a goto parancsok használatával. Az „**A**” változó kezdőértékét InputBox segítségével adja meg. Figyeljen arra, hogy az Inputbox paramétereit úgy adja meg, hogy az ábrán látható ablakot eredményezze. A program a táblázat fejlécét, valamint az „**A**”, „**N**” értékét az alábbi ábrán látható módon Excel cellákba írja ki, míg „**Program vege**” szöveget a MsgBox paranccsal a képernyőre írja ki. A program megírása során használt változók integer típusúak legyenek. A maradékosztáshoz használja a „**Mod**” parancsot. (11 pont)
   Írja át a programot úgy, hogy a ciklust létrehozó if - then és a goto parancsok helyett, Do - Loop While ciklusutasítást, az if - then - else szerkezet helyett pedig a Select Case szerkezetet használja. (4 pont)

	A	B	C
1			
2	No.	Szam	
3	1	70	
4	2	72	
5	3	24	
6	4	8	
7	5	9	
8	6	3	
9	7	1	
10			

3.3.4.2 1. feladatsor, B csoport

Név: **Tankör:** **Diger-1, B csoport – 2011 ősz**

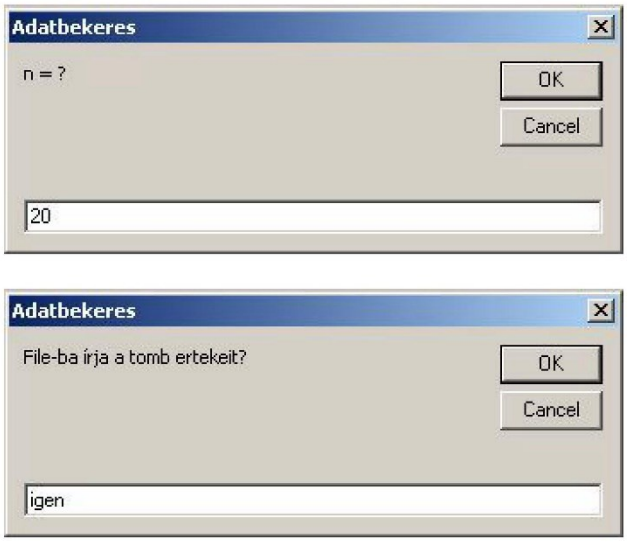
1. A megadott blokkdiagram alapján készítse el a „B1a” programot a For - Next és a Do - Loop While ciklusutasítások használatával. Az „**Max**” változó kezdőértékét InputBox segítségével adja meg. Figyeljen arra, hogy az Inputbox paramétereit úgy adja meg, hogy az ábrán látható ablakot eredményezze. A program a táblázat fejlécét, valamint a 3. sor celláinak értékét az alábbi ábrán látható módon írja ki, míg „**Program vege**” szöveget a MsgBox paranccsal a képernyőre írja ki. A program megírása során használt változók integer típusúak legyenek. A véletlenszerű 0, 1, és 2 értékek létrehozásához a „3 * Rnd” és az egészosztás parancsokat használja. (13 pont)
   Írja át a programot, az if - then - else szerkezet helyett a Select Case szerkezetet használja! (2 pont)

	A	B	C
1			
2	A>B	A=B	A<B
3	24	15	21

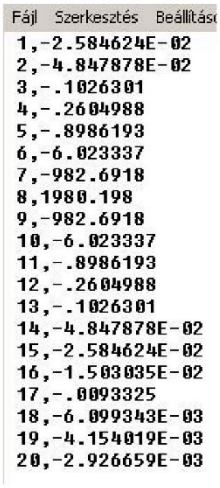
3.3.4.3 2. feladatsor, A csoport

Név: **Tankör:** **Diger-2, A csoport – 2011 ősz**

1. Írja meg az „A1” programot, amely először az Inputbox utasítás segítségével beolvassa „n” változó értékét (figyeljen arra, hogy az Inputbox paramétereit úgy adja meg, hogy az az ábrán látható ablakot eredményezze!), majd definiálja a „Tomb” nevű „n”-méretű dinamikus tömböt. Utána „i” = 1-től egyesével „n”-ig haladva számolja ki a „Tomb” változó „i.” eleméhez a megfelelő $2\cdot f(i)-(f(i-1)+f(i+1))$ értéket, ahol f(i) értékét a $\frac{10}{(i-8)^2+0{,}01}$ kifejezés adja meg. A kifejezés kiszámolására írja meg az „f” nevű függvényt. A program határozza meg „Tomb” legnagyobb értékét („Max” változó), „i”-t, „Tomb” értékeit és a legnagyobb értéket az ábrán látható módon írja ki a munkalapra (a fejléc is programozandó), valamint a felhasználó döntése esetén (Inputbox) „i” és „Tomb” értékeit az ábrán látható módon a „Tomb.txt” file-ba írja ki. A főprogram megírása során az „i”, „n”, „Max”, „Tomb” és „Dontes” (utóbbi string) változókat használja, „f” függvény megírása során (amennyiben szükséges) további változókat is alkalmazhat. (15 pont)



	A	B	C	D	E
1	i	Ertek		Max =	1980,198
2	1	-0,02585			
3	2	-0,04848			
4	3	-0,10263			
5	4	-0,2605			
6	5	-0,89862			
7	6	-6,02334			
8	7	-982,692			
9	8	1980,198			
10	9	-982,692			
11	10	-6,02334			
12	11	-0,89862			
13	12	-0,2605			
14	13	-0,10263			
15	14	-0,04848			
16	15	-0,02585			
17	16	-0,01503			
18	17	-0,00933			
19	18	-0,0061			
20	19	-0,00415			
21	20	-0,00293			
22					



3.3.4.4 2. feladatsor, B csoport

Név: **Tankör:** **Diger-2, B csoport – 2011 ősz**

1. Írja meg az „B1” programot, amely először az Inputbox utasítás segítségével beolvassa „n” változó értékét (figyeljen arra, hogy az Inputbox paramétereit úgy adja meg, hogy az az ábrán látható ablakot eredményezze!), majd definiálja a „Matrix” nevű „n × n”-méretű dinamikus tömböt. Utána a „Matrix” minden egyes (i, j) eleméhez (i: sorindex, j: oszlopindex) számolja ki a $g(i,i)+g(i,j)-g(j,i)+g(j,j)$ értéket, ahol $g(x,y)=x^2+2\cdot y$. A g(x,y) kifejezés kiszámolására írja meg a „g” nevű függvényt. A program a határozza meg „Matrix” elemeinek összegét („Osszeg” változó), valamint „Matrix” értékeit a munkalapra, míg „Osszeg” értékét a „Matrix.txt” file-ba írja ki az ábrákon látható módon. (12 pont)
   A programot írja át úgy, hogy a „g” függvény helyett a „g2” szubrutint alkalmazza, mely a $g(i,i)+g(i,j)-g(j,i)+g(j,j)$ értékét a kifejezés tagjainak összevonása után kapott $2\cdot i^2+4\cdot j$ összefüggéssel határozza meg. (3 pont)
   A főprogram megírása során az „i”, „j”, „n”, „Matrix” és „Osszeg” változókat használja, „g” függvény és „g2” szubrutin megírása során (amennyiben szükséges) további változókat is alkalmazhat.

	A	B	C	D	E	F	G	H
1	6	10	14	18	22	26	30	34
2	12	16	20	24	28	32	36	40
3	22	26	30	34	38	42	46	50
4	36	40	44	48	52	56	60	64
5	54	58	62	66	70	74	78	82
6	76	80	84	88	92	96	100	104
7	102	106	110	114	118	122	126	130
8	132	136	140	144	148	152	156	160

3.3.4.5 3. feladatsor, A csoport

Név: **Tankör:** **Diger-3, A csoport – 2011 ősz**

1. Írja meg az „A1” programot, amely először az Adatok.txt file tartalmát (pl.: 1, 20, 36, 15, 96, 48, 55, 72, 28, 88, 92, 35, 14, 19, 62, 49, 76, 32, 99, 44) olvassa be a „Tomb” nevű dinamikus tömbbe (használja az EOF kifejezést!). Utána az Inputbox utasítások segítségével olvasson be alsó és felső határértéket az „also” és a „felso” változókba (figyeljen arra, hogy az Inputbox paramétereit úgy adja meg, hogy az az ábrán látható ablakot eredményezze!). Ezután a program keresse ki a „Tomb” dinamikus tömb elemei közül azokat, amelyekre igaz, hogy also <= Tomb(i) <= felso, összesítse, hány megfelelő számot talált, majd az eredményt az alábbi ábrán látható módon írja ki a munkalapra. Amennyiben a program nem talál a „Tomb” dinamikus tömb elemei közül feltételeknek megfelelőt, ezt MsgBox ablakkal is hozza a felhasználó tudomására! A felsorolt változók legyenek integer típusúak! A feladat megoldása során (amennyiben szükséges) további változókat is alkalmazhat (pl.: i, j, k, talalat, stb.) (15 pont)

	B	C	D	E
1	Ertek		Also hata	50
2			Felso hata	54
3			Talalat =	0
4				
5				

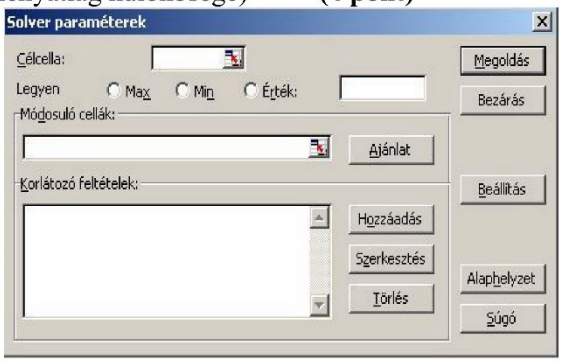
	A	B	C	D	E
1	No.	Ertek		Also hatar =	15
2	1	20		Felso hatar =	26
3	2	15		Talalat =	3
4	3	19			
5					

3.3.4.6 4. feladatsor, A csoport

Név: **Tankör:** **Diger-4, A csoport – 2011 ősz**

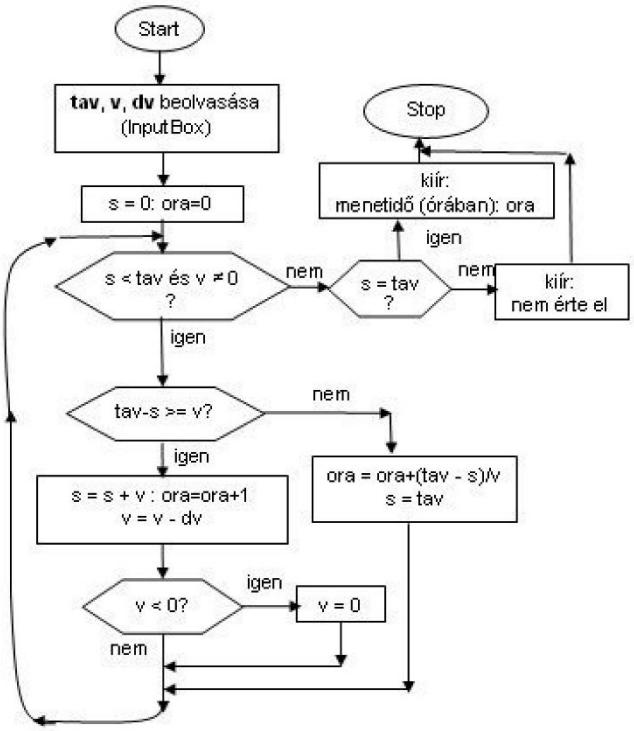
1. A 2010. 09. 28-i Zalaegerszeg-Videoton labdarúgó-mérkőzés eredményére fogadtunk úgy, hogy az elérhető fogadóirodákból kiválasztottuk azt a hármat, amely a mérkőzés adott kimenetele (1 ; x ill. 2) esetén a tétből a legnagyobb szorzóval (3,20 ; 3,59 ill. 2,40) számított nyereményt fizeti. Összesen 10000 forintot tettünk meg a három kiválasztott fogadóirodánál oly módon, hogy célunkhoz – függetlenül a mérkőzés végeredményétől ugyanakkora nyereményt kívántunk elérni – az összesen 10000 Ft-nyi tét felosztását Solver-rel számítottuk. Milyen képleteket használtunk a B5, D3-D5, D7, F3-F5 és F7 cellák értékeinek kiszámolásához? Melyek voltak a módosuló cellák, a célcella, illetve mire állítottuk be a „Legyen” értékét? (A hiba az adott nyeremény és a nyereményátlag különbsége) **(4 pont)**

	A	B	C	D	E	F
1						
2		Tét	Szorzó	Nyeremény		Hibanégyzet
3	Zalaegerszeg		3,20			
4	Döntetlen		3,59			
5	Videoton		2,40			
6						
7			Átlag =		Összesen =	

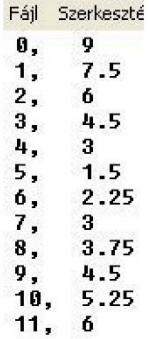


2. Írjon VBA programot a mellékelt blokkdiagram alapján. A programban használjon Do - Loop ciklusutasítást, a **tav**, **v** és **dv** változók kezdőértékét InputBox segítségével adja meg, és minden változót deklaráljon Single-ként. **(8 pont)**

3. Készítsen VBA **Function**-t, melynek neve legyen **fajlagos,** formális paraméterei **r** és **m** (egy henger alaplapjának sugara és a henger magassága) **Double** típusúak. A **Function** neve vegye fel a henger **A/V** fajlagos felületének (**Double** típusú) értékét, ahol $\mathbf{A}= 2\mathbf{r}^2\pi+2\mathbf{r}\pi\mathbf{m}$ és $\mathbf{V}=\mathbf{r}^2\pi\mathbf{m}$. A **Function**-ban deklaráljon **pi** nevű, **Double** típusú változót π értékének tárolására: **pi** = 4 * atn (1)
A főprogramban olvastassa be Inputbox segítségével egy For-Next cikluson belül 10 henger **r** és **m** adatait, és számítsa a **fajlagos Function** hívásával a megfelelő fajlagos felületeket. Az **r**, **m** és fajlagos felület értékeket irassa ki egy Excel munkalap első 10 sorának A, B és C oszlopaiba. **(6 pont)**



4. Készítsen VBA programot, amely először megnyitja az „Adatok.txt” fájlt, majd a benne lévő adatpárokat beolvassa az (1. oszlop) **x Integer** típusú illetve az (2. oszlop) **y Single** típusú egyindexes dinamikus tömbökbe. Az **n** (**Integer** típusú) változóban tárolja a beolvasott adatok számát, amely egyben az **x** tömb legnagyobb értékű eleme. Az **a** és **b** változókba InputBox-szal olvastasson be két egész számot, melyekre vizsgálja $\mathbf{0 \le a < b \le n}$ feltételek teljesülését. (Ha nem teljesülnek, a beolvastatást a program „**a** nem jó” ill. „**b** nem jó” üzenetet adva, a feltételek teljesüléséig ismételje.) Ezután a program számítsa ki az $S = \sum_{k=a+1}^{b} \frac{y(k)+y(k-1)}{2}$ összeget, majd írja ki **a**, **b** és **S** értékét. A program minden változóját deklarálja! **(12 pont)**



3.3.5 2012 őszi feladatsorok

3.3.5.1 1. feladatsor, A csoport

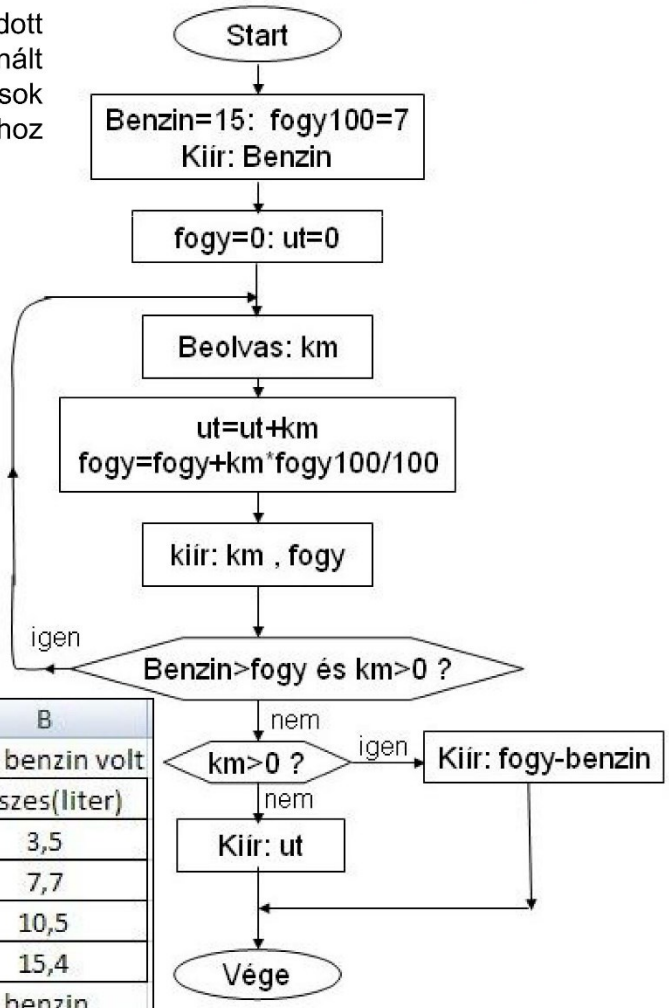
A	**Z1 (2012. okt.30.)**	Név:	tk:

1. Írja be az alább (jobbra) látható Excel „Munkalap” megfelelő celláiba, hogy mit ír ki az alább (balra) megadott VBA program. (4,5 p.)

```
Sub F1_A()
Dim n As Integer, k%, b#
    n = (3 ^ 2 + 10) / 3: k = 2
vissza:
    b = (k ^ 2 + 1) / 2
    Cells(k, 1) = k: Cells(k, 2) = b
    k = k + 1
  If k < n Then GoTo vissza
  Cells(1, 1) = "k": Cells(1, 2) = "b"
  Cells(k, 1) = "n=": Cells(k, 2) = n
End Sub
```

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

2. Készítsen VBA programot az itt megadott blokkdiagram alapján. A programban használt összes változót deklarálja, a beolvasások InputBox-szal történjenek, a „hurok” leírásához hátul tesztelt ***Do - Loop*** ciklust használjon.



A programban a kiiratások (az oszlopok fejlécét és az egyéb szöveges részeket is beleértve) az alábbi ábrák szerint történjenek. Az egyes ábrákhoz tartozó beolvasott ***km*** értékek mindkét ábrán az ***A*** oszlopban láthatók. A baloldali ábra olyan kiiratást mutat, amikor a ***km*** utolsó beolvasott ***0*** értéke okozza a ciklusból való kilépést. A jobboldali ábra a ciklusból való olyan kilépést mutat, amikor a ***km*** utolsó beolvasott értékéhez tartozó benzin szükséglet már nem áll rendelkezésre. (10,5 p.)

	A	B
1	15	liter benzin volt
2	Rész(km)	Összes(liter)
3	50	3,5
4	60	7,7
5	40	10,5
6	0	10,5
7	150	km utat tettél
8		Megérkeztél!

	A	B
1	15	liter benzin volt
2	Rész(km)	Összes(liter)
3	50	3,5
4	60	7,7
5	40	10,5
6	70	15,4
7	0,4	liter benzin
8	kellene még a célhoz	

3.3.5.2 1. feladatsor, B csoport

Név:	tk:	Z1 (2012. okt.30.)	B

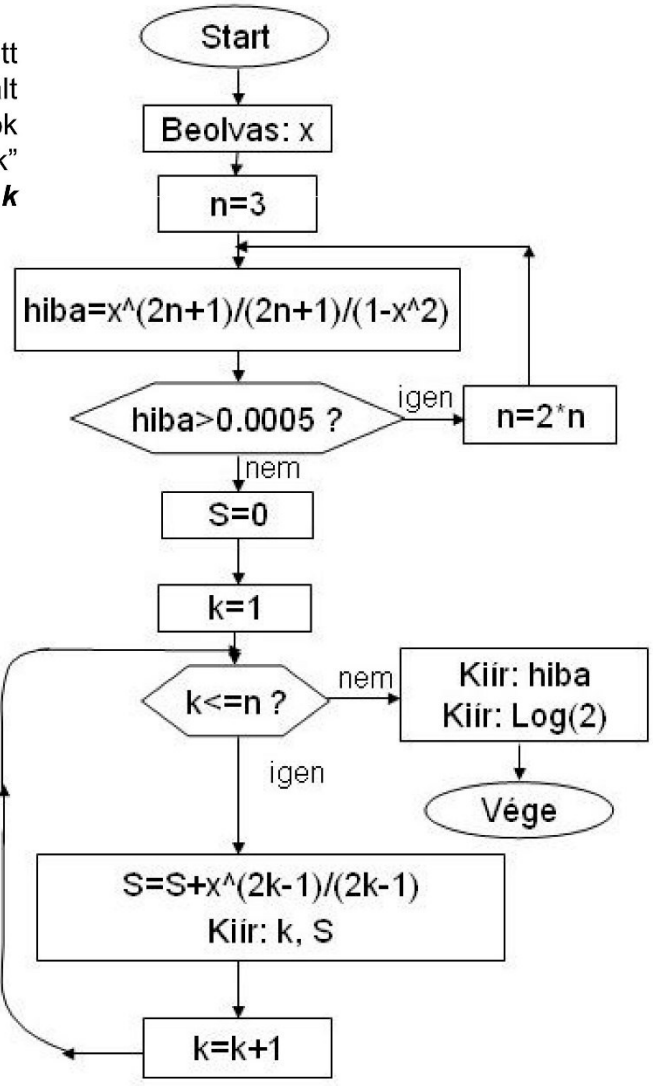
1. Írja be az alább (jobbra) látható Excel „Munkalap” megfelelő celláiba, hogy mit ír ki az alább (balra) megadott VBA program. (4,5 p.)

```
Sub F1_B()
Dim x As Double, y#, m%
   x = 1 + 3 * 0.2 ^ 2: m = 5
   Cells(1, 1) = "x=": Cells(1, 2) = x
ujra:
     m = m - 1
     y = 3 * (m + 1) / 2
     Cells(2, m) = m: Cells(3, m) = y
  If m - x > 1 Then GoTo ujra
   m = m - 1
 Cells(2, m) = "m": Cells(3, m) = "y"
End Sub
```

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

2. Készítsen VBA programot az itt megadott blokkdiagram alapján. A programban használt összes változót deklarálja, a beolvasások InputBox-szal történjenek, a (második) „hurok” leírásához elől tesztelt ***Do – Loop,*** (vagy ***k*** ciklusváltozójú ***For – Next***) ciklust használjon.

A programban a kiiratások (az oszlopok fejlécét és az egyéb szöveges részeket is beleértve) az alábbi ábra szerint történjenek. (Az ábra az x=0,6 input értékhez tartozó kiiratásokat mutatja). (10,5 p.)



	A	B
1	k	S
2	1	0,6
3	2	0,672
4	3	0,6876
5	4	0,6916
6	5	0,6927
7	6	0,6930
8		
9	hiba=	1,57E-04
10	ln2=	0,6931

3.3.5.3 2. feladatsor, A csoport

A	**Z1 (2012. nov.29.)**	Név:	tk:

A jobbra látható ***IskTam.txt*** fájl az ***Iskolák*** szöveg után a támogatott iskolák számát, nevét és az adományokból való részesedési hányadát tartalmazza, majd a ***támogatók*** szöveget a támogatók száma, neveik, és az egyes nevek mellett az első ill. második félévi adomány ezer Ft-ban megadott összegei követik. Az alább megadott feladatokat úgy oldja meg, hogy a program olyan (az ***IskTam.txt*** fájllal azonos szerkezetű) input fájl esetén is helyes eredményt adjon, ahol az iskolák és a támogatók száma más, de a támogatók száma <= 100.

```
Iskolák, 3
Focisuli, Képző, Alap
0.4, 0.25, 0.35
támogatók, 10
Balogh, 22,  0
Cecey,  15,  0
Daróci,  0, 14
Fodor,  25,  0
Kiss,    0, 35
Nagy,   15, 13
Papp,    0, 24
Szabó,   0, 43
Tóth,   32,  0
Zalai,  15,  0
```

1. Készítsen ***Tamogat*** nevű VBA programot a fájl adatainak beolvasására, és az adatokból az egész évi összes támogatás, valamint ezen összegből az egyes iskolákra jutó részek kiiratására az alább látható táblázat ***A, B, C*** oszlopa szerinti formában.

Az iskolák nevét egy ***String*** típusú, ***Isk*** nevű tömbbe, az egyes iskoláknak jutó támogatási hányad értékeket egy ***Double*** típusú, ***resz*** nevű tömbbe olvastassa. Ez a két tömb dinamikus indexhatárú legyen. A k-adik támogató első ill. második féléves adományait az ***eFt%(100,2)*** –ként deklarált tömb ***(k,1)*** ill ***(k,2)*** indexeknek megfelelő elemébe olvastassa. A programban használt összes változót deklarálja. **(7,5 pont)**

(Ha az ***Isk*** és ***resz*** tömböket nem dinamikus indexhatárral adja meg, a feladatra max. **6,5 pont**ot kaphat.)

	A	B	C	D	E	F	G	H	I
1	Teljes érték (EFt)		253		1.félévi támogatók			2.félévi támogatók	
2	Az egyes iskolák része(Ft)				Balogh	22		Daróci	14
3	Focisuli	Képző	Alap		Cecey	15		Kiss	35
4	101200	63250	88550		Fodor	25		Nagy	13
5					Nagy	15		Papp	24
6					Tóth	32		Szabó	43
7					Zalai	15		Összes(Eft)	129
8					Összes(Eft)	124			

2. Készítsen ***FelevFt*** nevű paraméteres ***Sub***-ot egy adott félév adományozói nevének és adományainak kiiratására, valamint az adományok összegzésére, és ezen összegek kiiratására.

A ***FelevFT*** formális paraméterei legyenek az ***Integer*** típusú ***melyik***, ***n*** és ***osz***, a ***String*** típusú ***Kitol*** tömb, valamint az ***Integer*** típusú ***Mennyi*** tömb, melyek rendre az adott félév számjelének (1 vagy 2), a támogatók (teljes) számának, a kiiratás kezdő oszlop-számának, valamint a neveket ill. adományokat tartalmazó tömböknek felelnek meg.

Hívja meg a ***FelevFt Sub***-ot a ***Tamogat*** program kétszer, úgy, hogy az egyes hívások eredményül a fenti táblázat ***E-F*** ill. ***H-I***. oszlopait adják. **(7,5 pont)**

3.3.5.4 2. feladatsor, B csoport

Név:	tk:	Z1 (2012. nov.29.)	B

A jobbra látható ***Pogacsa.txt*** fájl első sorában a két gyártott pogácsa nevét, majd a ***Figyelt anyag (kg/kg)*** szöveg után a figyelt alapanyagok számát, neveit és az egyes pogácsa fajták 1 kg-jához szükséges mennyiségeit adja meg. Ezután a ***Rendelések(kg)*** szöveget az egyes pogácsa fajták napi rendelési értékei követik. Az alább megadott feladatokat úgy oldja meg, hogy a program olyan (a ***Pogacsa.txt*** fájllal azonos szerkezetű) input fájl esetén is helyes eredményt adjon, ahol a figyelt anyagok száma és a rendelések napjainak a száma más, de a figyelt anyagok száma <= 5, és a rendelések napjainak a száma <=30.

```
Sajtos, Túrós
Figyelt anyag (kg/kg), 3
Liszt,  0.7, 0.6
Vaj,    0.1, 0.15
Tejföl, 0.1, 0.15
Rendelések(kg)
10, 0
 0, 10
10, 10
15, 20
12, 14
10, 20
```

1. Készítsen ***Pogi*** nevű VBA programot a fájl adatainak beolvasására, és az adatokból az egyes napokra vonatkozó alapanyag fogyasztás kiszámítására és kiiratására az jobbra látható táblázat ***A, B, C, D*** oszlopa szerinti formában. A pogácsa fajták nevét a ***PogiNev$(2)*** – ként, a rendelések kg értékeit a ***rend#(30,2)*** – ként deklarált tömbökbe olvastassa, ez utóbbinál a második index 1 értéke az első (a Sajtos), 2 értéke a második (a Túrós) pogácsa rendelési kg-jára vonatkozik. **(8 pont)**

	A	B	C	D	E	F	G
1	Napi fogyás(kg)					Összes kg 6 nap alatt	
2	nap	Liszt	Vaj	Tejföl		Sajtos pogácsa:	57
3	1	7	1	1		Túrós pogácsa:	74
4	2	6	1,5	1,5			
5	3	13	2,5	2,5			
6	4	22,5	4,5	4,5			
7	5	16,8	3,3	3,3			
8	6	19	4	4			

2. Készítsen ***OsszFogy*** nevű, ***Integer*** típusú ***Function***-t az adatfájlban megadott napokon összesen megrendelt, adott fajtájú pogácsa mennyiségének a kiszámítására. A ***Function*** neve az adott fajtájú pogácsa kg-ban számított mennyiségét vegye fel.
A ***Function*** formális paraméterei legyenek az ***Integer*** típusú ***a*** és ***n***, valamint a ***Double*** típusú ***x*** tömb, melyek rendre a pogácsaneveket tartalmazó tömb index-értékének, az összes rendelési nap számának, valamint a rendelési kg-okat tartalmazó tömbnek felelnek meg.
Hívja meg az ***OsszFogy Function***-t a ***Pogi*** program kétszer úgy, hogy az egyes hívások eredményül a fenti táblázat ***G2 ill. G3*** cellákat adják. A ***Pogi*** program hajtsa végre az ***F*** oszlopban látható 3 kiírást is. **(7 pont)**

3.3.5.5 3. feladatsor, A csoport

Név: **Tankör:** **Diger-3, A csoport – 2012 ősz**

1. Írja meg az „A1” programot, amely
a) először az Utazas.txt file-ban lévő ismeretlen számú adatpárokat olvassa be a „Varos” (sztring típusú) és „Ut” (integer típusú) nevű dinamikus tömbökbe a városok nevét, illetve a menettérti autóút hosszúságát (használja az EOF kifejezést!), miközben a „db” változóba rögzíti a beolvasott adatpárok számát,
b) majd az Inputbox utasítás segítségével bekéri a „Uticel” változóba (sztring) a keresendő város nevét.
c) Ezután a program keresse ki a „Varos” dinamikus tömb elemei közül azokat, amelyekre igaz, hogy Varos(i) = Uticel („i” ciklusváltozó). A program a „Celszam” (integer) változóban tárolja a találatok számát, illetve Uticelossz (integer) változóban rögzítse az adott városba történt utazások számának megfelelő összes megtett km értékét.
d) Végezetül az alábbi ábrán látható módon írja ki az eredményt. (11.5 pont)

	A	B
1	Utazások száma:	18
2	Célpont:	Pécs
3	Célpontutazások (szam):	4
4	Célpontutazások (km):	1584

Fájl Szerkesztés Beállítá
Szolnok, 194
Győr, 247
Pécs, 394
Nyíregyháza, 490
Veszprém, 221
Veszprém, 219
Kaposvár, 378
Pécs, 397
Eger, 255
Debrecen, 452
Győr, 245
Tatabánya, 112
Pécs, 396
Veszprém, 220
Szeged, 341
Pécs, 397
Szeged, 343
Eger, 257

2. a) Írja meg a „A2” program azon részét, mely az „y” (double) változójának értékadásakor meghívja az „Faktorialis” függvényt, melyhez egyetlen paraméter, az „x” (integer) változó tartozik („x” és „y” deklarálása nem szükséges). Az „y” értékét MsgBox paranccsal jelenítse meg.
b) Írja meg a double típusú „Faktorialis” függvényt, mely a „A2” program „x” paraméterét veszi át a függvény „xbe” integer típusú változójába és kiszámítja a

$$\prod_{i=xbe}^{1} i = xbe \cdot (xbe-1) \cdot (xbe-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$$ értékét („i” ciklusváltozó),

amit majd átad a főprogram „y” paraméterének. (3.5 pont)

3.3.5.6 3. feladatsor, B csoport

Név: **Tankör:** **Diger-3, B csoport – 2012 ősz**

1. Írja meg az „B1” programot, amely
a) először az Vasarlas.txt file-ban lévő ismeretlen számú adatpárokat olvassa be a „Aru” (sztring típusú) és „Kiadas” (integer típusú) nevű dinamikus tömbökbe a vásárolt árúk nevét, illetve a fizetett összeget (használja az EOF kifejezést!), miközben a „db” változóba rögzíti a beolvasott adatpárok számát,
b) majd az Inputbox utasítás segítségével bekéri a „Tetel” változóba (sztring) a keresendő tétel nevét.
c) Ezután a program keresse ki a „Aru” dinamikus tömb elemei közül azokat, amelyekre igaz, hogy Aru(i) = Tetel („i” ciklusváltozó). A program a „Tetelszam” (integer) változóban tárolja a találatok számát, illetve „TetelFt” (integer) változóban rögzítse az adott tétel vásárlására költött összeget.
d) Végezetül az alábbi ábrán látható módon írja ki az eredményt. (11.5 pont)

	A	B
1	Kiadások száma:	20
2	Vásárlás (tétel):	élelmiszer
3	Vásárlás (szam):	3
4	Vásárlás (Ft):	12185

Fájl Szerkesztés Beállítások
BKV bérlet, 9800
újság, 175
élelmiszer, 2870
telefonszámla, 3200
tisztítószer, 1570
DVD film, 990
írószer, 1255
menza, 890
élelmiszer, 4150
könyv, 2480
fogkrém, 285
újság, 310
menza, 1120
DVD film, 1980
Főgáz számla, 2960
élelmiszer, 5165
Elmü számla, 5690
könyv, 1565
menza, 920
mozijegy, 1290

2. Írja meg a „B2” program azon részét, mely az „y” (double) változójának értékadásakor meghívja az „Egyperiksz” függvényt, melyhez egyetlen paraméter, az „x” (integer) változó tartozik („x” és „y” deklarálása nem szükséges). Az „y” értékét MsgBox paranccsal jelenítse meg.
b) Írja meg a double típusú „Egyperiksz” függvényt, mely a „B2” program „x” paraméterét veszi át a függvény „xbe” integer típusú változójába és kiszámítja a

$$\sum_{i=1}^{xbe} \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{xbe-2} + \frac{1}{xbe-1} + \frac{1}{xbe}$$ értékét („i” ciklusváltozó),

amit majd átad a főprogram „y” paraméterének. (3.5 pont)

3.3.5.7 4. feladatsor, A csoport

Név: **Tankör:** **Diger-4, A csoport – 2012 ősz**

1. Egészítse ki a baloldali programrészletet a jobb oldali oszlopban lévő 1-16. számú részletekkel. (8 pont)

```
Sub Feladat1___

___ i%, j___, x___

___:
x = ___("x=?")
If x ___ -3.5 ___x > 3.5 Then ___

For i = 1 To 4
   Cells(1, i + 1) = i
   Cells(i + 1, 1) = i
   For j = 4 To 1 ___ -1
      If i > j Then
         Cells(i + 1, j + 1) = i * x + j
         ___j > i ___
         Cells(i + 1, j + 1) = i * x - j
         ___
         Cells(i + 1, j + 1) = i * x
      ___
   ___
___

End Sub
```

1) ElseIf
2) ujra
3) Step
4) Next j
5) ()
6) InputBox
7) %
8) Next i
9) GoTo ujra
10) Dim
11) Then
12) !
13) Else
14) Or
15) End If
16) <

2. Írja meg a Feladat2 VBA programot, amely:
a) először az ábrán látható munkalapról beolvassa a „Hitel”, „Kamat” és „Reszlet” változókba (mind double) a felveendő hitel nagyságát, havi kamatát és a havi törlesztő részletet. A „Hitel” változó értékét adja át a „Maradek” változónak (double),
b) leellenőrzi, hogy a havi törlesztés meghaladja-e hitel havi kamatterhét. Amennyiben nem, akkor MsgBox üzenettel jelezze, hogy a hitel nem visszafizethető ilyen alacsony havi törlesztéssel.
c) ha a hitel visszafizethető, a „Maradek” változó értékét növelje meg a havi kamatteherrel és vonja le a törlesztőrészletet. Majd a „Torlesztes” változó (double) értékét pedig növelje meg a törlesztő összeg értékével. Ezt a ciklust addig csinálja, amíg a „Reszlet” nagyobb nem lesz a „Maradek” változónak kamatteherrel megnövelt értékénél. A program közben számolja a törlesztési hónapok számát („Honap” változó, integer). Ha a havi részlet nagyobb a „Maradek” változónak kamatteherrel megnövelt értékénél, a „Torlesztes” változó értékéhez csak ezt az összeget adja hozzá.
d) végezetül az eredményeket a munkalapon látható módon jelenítse meg (a „Különbség” a törlesztett és a felvett összeg különbségét jelenti). (7 pont)

	A	B
1	Hitel (Ft) =	1000000
2	Havi kamat (%)=	1
3	Részlet (Ft) =	20000
4		
5	Időtartam (hónap) =	69
6	Törlesztés (Ft) =	1393237
7	Különbség (Ft) =	393237

Változók és jelentésük:
Hitel: a teljes felvett pénz mennyiség (double),
Kamat: a havi kamat (double),
Reszlet: a havi törlesztés összege, vagyis a havi befizetés (double),
Torlesztes: a visszafizetett pénz összege (double),
Maradek: a teljes, még vissza nem fizetett pénz összege (double),
Honap: hónapok száma, amelyekben befizetés történt (integer).

3. Írja meg a Feladat3 VBA programot, amely:

a) a „Kartya.txt” file-ból (jobboldali ábra) a „Nevek” tömbbe (index: 4, string típusú) beolvassa a „Fekete macska” kártyajátékban résztvevő személyek nevét, majd a „Pontok” 4*20-as tömbbe (integer) a partikban szerzett pontok összegét, miközben az „osztas” változóban (integer) számolja meg a leosztások (partik) számát,

b) a neveket, a pontokat, valamint a játékok számát az alsó ábrán látható módon kiírja egy Excel munkalapra,

c) a felhasználótól a „J1” és „J2” változókba (integer) bekéri két játékos számát (pl.: Tibi: 1, Évi: 2),

d) meghívja „Kulonbseg” szubrutint, melynek a főprogram átadja a „Pontok”, „osztas”, „J1” és „J2” változókat a szubrutin „Pontokbe” (tömb), „osztasbe”, „J1be” és „J2be” változóiba (mind integer) és „Min” és „Max” (mindkettő integer) változókat, melyekbe a szubrutin a kiszámolt „Minki” és „Maxki” (mindkettő integer) változókat adja vissza. A „Kulonbseg” szubrutin a két játékos pontjai közötti legnagyobb eltéréseket határozza meg, ezeket rögzíti a „Minki” és „Maxki” változókban.

e) „J1” és „J2” játékosok nevét, valamint a köztük lévő legnagyobb különbségeket az ábrán látható módon írja ki az Excel munkalapra (a -14 a 12. leosztás, a 9 a 2. leosztás után alakult ki). A cellákba történő kiíráskor a sorindexet az „osztas” változó függvényében adja meg!

A programozás során használhat még ciklusváltozókat (i, j, k, mind integer), illetve a „Kulonbseg” szubrutinban lokális változókat is, ha szükségesnek tartja. (15 pont)

	A	B	C	D	E
1		Tibi	Évi	Zoli	Kati
2	1	14	6	6	0
3	2	18	9	19	6
4	3	19	24	25	10
5	4	20	29	40	15
6	5	25	32	55	18
7	6	28	32	58	38
8	7	28	34	60	60
9	8	28	40	62	78
10	9	32	42	76	84
11	10	46	44	86	84
12	11	48	50	90	98
13	12	52	66	95	99
14	13	66	68	103	101
15					
16	1. játékos (Jat1): Tibi				
17	2. játékos (Jat2): Évi				
18	Jat1-Jat2 legnagyobb pontkülönbségek:				
19	-14	9			