

Számítástechnika BSc

(Biomérnöki és Vegyészmérnöki Szak kötelező tantárgya)



VBA Programozási ismeretek

A Visual Basic for Application programozás tantárgyi követelményeinek összefoglalója. A programozás alapjai, alaki követelményei, felépítése, parancsai, eljárásai, fogalmai.

Összeállította: Rippel Endre, Simon András

Budapesti Műszaki és Gazdaságtudományi Egyetem

Vegyészmérnöki és Biomérnöki Kar

Szervetlen és Analitikai Kémia Tanszék

2013

1 Tartalomjegyzék

1	Tartalomjegyzék.....	2
2	Tárgyajánló.....	5
3	A számítástechnika úttörői.....	7
3.1	Neumann János (Neumann János Lajos)	7
3.2	Kemény János György.....	7
3.3	John Barden	9
3.4	Gróf András (Andy Grove, Andrew Steven Grove).....	9
3.5	Simonyi Károly (Charles Simonyi).....	10
3.6	Szentiványi Tibor.....	11
3.7	Jánosi Marcell.....	11
3.8	Dr. Náray Zsolt.....	11
4	Tantárgyi információk.....	12
5	VBA programozási ismeretek	13
5.1	Gondolatok a programozásról	13
5.2	Visual Basic for Applications (VBA) fejlesztő környezete	14
5.3	A VBA fejlesztőrendszer fogalmai	15
5.4	A Visual Basic programnyelve	16
5.5	Változók és konstansok használata.....	17
5.6	Alapvető adattípusok	19
5.7	Aritmetikai, relációs, logikai konverziós és szövegkezelő műveletek, matematikai függvények.....	20
5.8	A Visual Basic utasításai és parancsai.....	22
5.8.1	Az „Option ...” utasítások.....	22
5.8.2	Műveletek szöveges változóval.....	23
5.8.3	Cellaparancsok	25
5.8.3.1	Cella használata írásra - olvasásra	25
5.8.3.2	Cellák tartalmának törlése	26
5.8.4	Adatbevitel és adatkivitel objektumokkal	27
5.8.4.1	InputBox - A VBA program adatbeviteli objektuma.....	27
5.8.4.2	MsgBox - A VBA program adatkiviteli objektuma.....	28
5.8.5	Adatbevitel és adatkivitel *.txt fájl segítségével	28
5.8.6	Elágazó utasítások. Feltételes és feltétel nélküli utasítások.....	30
5.8.6.1	Go to utasítás	30
5.8.6.2	If ... then elágazó parancs	31
5.8.6.3	Select Case ... End Select elágazó parancs	33
5.8.7	Ciklusszervező utasítások.....	34
5.8.7.1	For ... Next ciklus	34
5.8.7.2	Do ... Loop ciklus	35
5.8.8	Programstrukturáló utasítások.....	37
5.8.8.1	Eljárás (Szubrutin).....	37
5.8.8.2	Függvény (Function)	38

6	Feladatok.....	40
6.1	Programozási mintagyakorlatok alapvető algoritmusokkal.....	40
6.1.1	„n” elemű numerikus tömb elemeinek összege.....	40
6.1.2	n x m elemű tömb elemeinek összege (összes, - oszlop, - sor, - átló).....	40
6.1.3	Tömbváltozó elemei közötti Min és Max értékek keresése.....	42
6.1.4	Sorfüggvények és Faktoriális számolás rutin eljárások.....	43
6.1.4.1	Sorfüggvény példa.....	43
6.1.4.2	Faktoriális számítása.....	44
6.1.5	Műveletek vektorokkal és tömbökkel.....	45
6.1.5.1	Vektorok összege, szorzata, hossza.....	45
6.1.5.2	Tömbök szorzása.....	46
6.1.6	Kisebb – nagyobb rendezés (Buborékredezés).....	47
6.2	Kidolgozott gyakorló feladatok.....	48
6.2.1	Program írása blokkdiagram alapján.....	48
6.2.1.1	Térfogatszámolás feladat.....	48
6.2.2	Gyakorlatok For - Next és Do - Loop ciklusok alkalmazására.....	52
6.2.2.1	Lottó feladat.....	52
6.2.2.2	Pithagorasz-féle számhármak meghatározása.....	54
6.2.2.3	Számkitaláló feladat.....	57
6.2.3	Gyakorlatok Function függvények és szubrutinok alkalmazására.....	59
6.2.3.1	Alapműveletek feladat.....	59
6.2.4	Fájlműveletek gyakorlatai.....	62
6.2.4.1	Kétfüggvényes feladat.....	62
6.3	Önállóan megoldandó gyakorló feladatok.....	64
6.4	Korábbi zárthelyi feladatsorok.....	65
6.4.1	2009 tavaszi feladatsorok.....	65
6.4.1.1	1. feladatsor.....	65
6.4.1.2	2. feladatsor.....	66
6.4.1.3	3. feladatsor.....	67
6.4.1.4	4. feladatsor.....	68
6.4.2	2009 őszi feladatsorok.....	69
6.4.2.1	1. feladatsor, A csoport.....	69
6.4.2.2	1. feladatsor, B csoport.....	70
6.4.2.3	2. feladatsor, A csoport.....	71
6.4.2.4	2. feladatsor, B csoport.....	71
6.4.2.5	3. feladatsor, A csoport.....	72
6.4.2.6	3. feladatsor, B csoport.....	73
6.4.2.7	4. feladatsor.....	74
6.4.3	2010 őszi feladatsorok.....	75
6.4.3.1	1. feladatsor, A csoport.....	75
6.4.3.2	1. feladatsor, B csoport.....	76
6.4.3.3	2. feladatsor, A csoport.....	77
6.4.3.4	2. feladatsor, B csoport.....	77
6.4.3.5	3. feladatsor, A csoport.....	78
6.4.4	2011 őszi feladatsorok.....	79
6.4.4.1	1. feladatsor, A csoport.....	79
6.4.4.2	1. feladatsor, B csoport.....	80
6.4.4.3	2. feladatsor, A csoport.....	81
6.4.4.4	2. feladatsor, B csoport.....	82
6.4.4.5	3. feladatsor, A csoport.....	83
6.4.4.6	4. feladatsor, A csoport.....	84
6.4.5	2012 őszi feladatsorok.....	85
6.4.5.1	1. feladatsor, A csoport.....	85
6.4.5.2	1. feladatsor, B csoport.....	86
6.4.5.3	2. feladatsor, A csoport.....	87

6.4.5.4	2. feladatsor, B csoport.....	88
6.4.5.5	3. feladatsor, A csoport.....	89
6.4.5.6	3. feladatsor, B csoport.....	89
6.4.5.7	4. feladatsor, A csoport.....	90
7	Függelék	92
7.1	<i>Rekordok és használatuk</i>	92
7.2	<i>A For Each – Next ciklus és alkalmazása.....</i>	92
7.3	<i>A Solver alkalmazása a programírás során</i>	92
7.4	<i>A „User Form” használata</i>	92

2 Tárgyajánló

Az elsőéves hallgatókban sokszor felvetődik a kérdés, hogy a vegyészmérnök- és biomérnök-hallgatóknak szükségük van-e a Visual Basic for Applications programozás tanulására.

A válasz egyértelműen: Igen! Ez igaz még akkor is, ha a programozás tanulása kezdőknek eleinte fáradtságos, de semmiképpen sem hátrányos, sőt sok szempontból nagyon is hasznos!

A programozás (bármilyen nyelven történjen is) logikus gondolkodásra nevel! Az objektumorientált nyelv, fejlesztő rendszerével végzett programozói munka hatékonyan fejleszti a rendszerszemléletű áttekintőképességet.

A fenti két képesség, (a logikus gondolkodás és rendszerszemléletű áttekintőképesség) önállóan nem tanulható, csak az ilyen gondolkodást igénylő feladatok megoldása közben sajátítható el. E képességek szintje nagyban determinálja a mérnöki tanulmányok elsajátításának eredményességét, de az igazi mérnöki munka, e képességek magas szintű művelése nélkül nem végezhető eredményesen.

Tehát a Számítástechnika tárgy témaválasztásával két előnyt egyesít:

- az egyetemi oktatásba belépő hallgatókat rögtön hozzásegíti a kíváncsi mérnöki gondolkodás felfedezéséhez és fejlesztéséhez.
- az oktatásra kijelölt objektumorientált Visual Basic programnyelv, minden Microsoft Excel szervesen beépített fejlesztő rendszere, így ez minden hallgatónak rendelkezésre áll és a hallgatók otthoni munkáját is lehetővé teszi.

E gondolatsort ajánljuk minden hallgatónknak! Még az oktatás előtt mindenki meggyőzheti magát arról, hogy a félév során fellépő „esetleges nehézségeinek” legyőzésével máris a mérnöki jövőjének gyakorlásához kap elengedhetetlen eszközt!

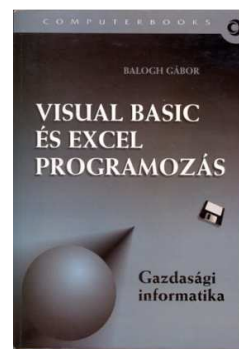
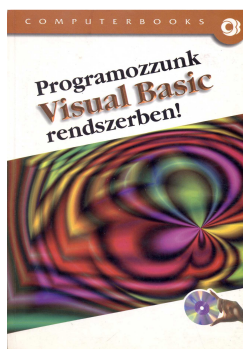
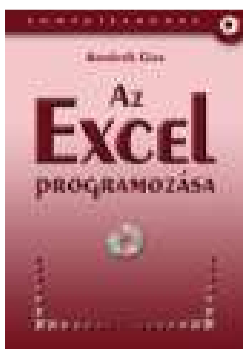
A Visual Basic fejlesztőrendszer az alapja a Microsoft Office programokba (Word, Excel, Access, PowerPoint) szervesen beépített programfejlesztő eszközöknek. Ezen összeállítás elsajátítása és gyakorlati alkalmazása a tárgy elvégzésének minimális feltétele, de nem helyettesíti a kiadott ajánlott irodalomjegyzék könyveinek tanulmányozását. Az összefoglaló csak segítséget ad, a szakkönyvek sokkal bővebb ismeretanyagának szűréséhez.

Ajánlott irodalom:

- Kovalcsik Géza: Az Excel programozása, ComputerBooks, Budapest, 2008, ISBN 963-618-332-5
- Balogh Gábor: Visual Basic és Excel programozás, ComputerBooks, Budapest, 2002, ISBN 963-618-229-9
- Kuzmina Jekatyerina, Dr. Tamás Péter, Tóth Bertalan: Programozzunk Visual Basic rendszerben, ComputerBooks, Budapest, 2006, ISBN 963-618-308-2
- Billo E. Joseph: Excel® for Chemists: A Comprehensive Guide, Wiley-VCH, New York – Chichester

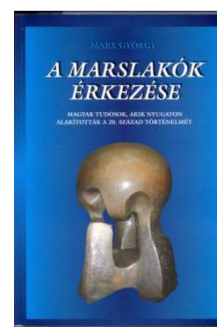
– Weinheim – Brisbane – Singapore – Toronto, 2001, ISBN 0-471-39462-9 (Paperback), 0-471-22058-2 (Electronic)

- Billo E. Joseph: Excel for scientists and engineers, John Wiley & Sons, Inc., Hoboken, New Jersey, ISBN: 978-0-471-38734-3



... és még egy ajánlat minden tanulmányait kezdő hallgatónknak:

Marx György: „A marslakók érkezése” című, 2000-ben írt könyve (Akadémia kiadó) bemutatja, hogy a 20. század robbanásszerű ipari - technikai fejlődését a Budapest iskoláiból „kirajzó” magyar tudósok milyen mértékben határozták meg.



Az informatika, a számítógép és számítástechnika, a félvezetők és a processzorgyártás, a mikrohullámú technika, az űrkutatás, a radioaktivitás, az atomkor (bombától - erőműig), a váltóáramú technika, a wolframszálas és kriptonégők, az autózástechika, a villamos vontatás, a repüléstechnika, a biológia és az orvostudomány, a filmgyártás (Hollywood) ... fejlődésének kezdete, mind - mind magyar nevekhez is kapcsolódik.

3 A számítástechnika úttörői

3.1 Neumann János (Neumann János Lajos)

(Budapest, Lipótváros, 1903. december 28. – Washington, 1957. február 8.) matematikus.



A Fasori Evangélikus gimnáziumban tanult. 1921-ben Neumann beiratkozott a budapesti Eötvös Lóránd Tudományegyetem matematika szakára. Egyetemi évei alatt sokat tartózkodott Berlinben, ahol Fritz Habertnél kémiát, Albert Einsteinnél statisztikus mechanikát és Erhardt Schmidtnél matematikát hallgatott. Neumann 1923-ban Zürichbe ment, hogy a zürichi Szövetségi Műszaki Egyetemen vegyészetet tanuljon. Vegyészmérnöki diplomáját 1925-ben szerezte, matematikából pedig egy évvel

később, 1926-ban doktorált Budapesten. 1930-ban meghívták vendégprofesszornak az Egyesült Államokba, Princeton-ba. Hamarosan az ottani egyetem professzora lett (1931), majd az újonnan megnyílt a princetoni Institute for Advanced Studies professzora (1933-1955) – John von Neumann néven – ahol a világ legkiválóbb tudósai gyűltek össze. A Los Alamos-i híres 5 magyar egyike, de az elektronikus számítógépek logikai tervezésében is kiemelkedő érdemeket szerzett. Megalkotta azt a gépstruktúrát, amit ma Neumann-elvnek, Neumann típusú számítógépnek neveznek:

- a kettes számrendszer alkalmazása,
- a szerkezet logikai irányítását, azaz a műveletek megkívánt sorrendjét legelőnyösebben egy kontrol/központ [processzor] végezheti. Ha azt akarjuk, hogy a szerkezet sokoldalú legyen, akkor az éppen tárgyalt probléma speciális utasításrendszerét [software] meg kell különböztetni a kontrol/központtól [hardware], amely elvégzezteti a géppel a speciális műveletet.

Tanácsadóként szerepelt az EDVAC tervezésénél. Ez volt az első olyan számítógép, amely a memóriában tárolta a programot is, 1944-től kezdték építeni, 1952-ben helyeztek üzembe.

Elméletét szabadalmaztatta, de szabad felhasználásra szétküldte az akkor ismert számítógépgyártóknak. A Neumann-elv alapján készülnek a mai számítógépek is.

3.2 Kemény János György

(Budapest, 1926. május 31. – New Hampshire, USA, 1992. december 26.), matematikus, számítástechnikus.



Kemény János, az 1970-es, 1980-as években az Egyesült Államokban Teller Edén kívül valószínűleg a legismertebb magyar-amerikai tudós volt.

Gimnáziumba a Berzsényi Dániel Gimnáziumba járt, de 1940 januárjában a család a hitleri Németország növekvő befolyása elől külföldre emigrált. Középiskolai tanulmányait New Yorkban fejezte be, majd a Princetoni Egyetemen végzett matemati-

kusként, és 1949-ben doktorált logikából. Előbb a Kenti Egyetem munkatársa lett. Jellemző volt rá, hogy autójára, a „LOGIC” (LOGIKA) rendszámot íratta rá. Majd 27 évesen meghívták a Dartmouthi Főiskolára matematika-professzornak, s két év múlva a Matematikai Intézet vezetője lett. 1962-ben ő javasolta az egyetemi számítógépközpont megépítését is, a központot végül 1966-ban adták át.

Kemény János azon gondolkozott, hogyan tegye a számítógépet hozzáférhetővé egyszerre több használó számára. Amíg a használó begépel, vagy a számítógép kigépel valamit, a processzor semmit sem tesz! Ezért Kemény bevezette az időbeosztás módszerét: minden használó saját terminálján dolgozik, a központi számítógép pedig beosztja processzorának munkaidejét a használók közt. A másodperc minden törtrészt kihasználják, mindegyik használó elégedett lehet, mert úgy érezheti: a központi gép vele foglalkozik. A processzor időbeosztását nem a használók intézik, hanem maga a központi számítógép. A számítógép-időbeosztás rendszerét először a Dartmouth Kollégium vezette be (1963).

Időbeosztásos rendszeréért Kemény János kapta az IBM legelső Robinson-díját (1991).

Abban az időben a FORTRAN volt a kutatók közt legelterjedtebb nyelv, de nem volt elég emberszabású. Kemény elhatározta, hogy egy interaktív nyelvet fejleszt ki, amelyik rögtön reagál a használó utasítására, így lehetővé teszi, hogy minden diák vagy felnőtt próba-szerencse alapon lépésről-lépésre építse föl, tapasztalja ki saját programját. Megfogalmazta elvárásait:

- A programozási nyelv legyen alkalmas eltérő célok kielégítésére.
- A nyelv kezdők számára is könnyű legyen.
- A magasabb szintű utasításokat csak később kelljen megtanulni.
- A használó és a gép között a nyelv legyen interaktív.
- Könnyen érthető hibajelzéseket adjon.
- A speciális gép-architektúra ismerete nélkül is lehessen használni.
- Óvja meg a használót a gép operációs rendszerének problémáitól.

Így született meg Kemény János és Tom Kurtz kezében 1964-ben a BASIC nyelv.

„Nem csak azért teremtettem meg a BASIC-et, hogy eggyel több számítógépes nyelv legyen. Azért csináltam, hogy a számítógép minden egyetemi hallgató (és minden diák) számára hozzáférhetővé váljék”. Az első BASIC program 1964. május 1-jén reggel 5 órakor futott.

(Kemény és Kurtz levédte a BASIC nevet, de mindenki szabadon használhatta-adaptálhatta a nyelvet. Így a BASIC a 20. század második felének legelterjedtebb programozási nyelve lett.)

Kemény János egyébként a ma közkedvelt elektronikus levelezés (e-mail) úttörője volt. Felesége egy 200 km távolságban levő főiskolán dolgozott. A két főiskola központi gépének összekapcsolásával létrejött az első „internet” amelyen keresztül levelezhettek.

3.3 John Barden

(Madison, Wisconsin, 1908. május 23. – Boston, 1991. január 30.) amerikai fizikus.

Wigner Jenő tanítványa, 1947-ben megalkotta a vákuum-elektroncsöveket felváltó tranzisztort, mely egy germánium kristály félvezető elem (mérete néhány mm²).

Az egyetlen ember, aki kétszer kapta meg a fizikai Nobel-díjat:

- 1956-ban a tranzisztor feltalálásáért William Bradford Shockley-lal és Walter Brattainnel együtt, és
- 1972-ben a konvencionális szupravezetés elméletért Leon Neil Cooperrel és John Robert Schriefferrel együtt.



3.4 Gróf András (Andy Grove, Andrew Steven Grove)

(Budapest, 1936. szeptember 2. –), vegyészmérnök, az Intel Corporation társalapítója.



Felsőfokú tanulmányait az Eötvös Loránd Tudományegyetem kémikus szakán kezdte. 1956-ban, a forradalom leverése után az Egyesült Államokba emigrált. 1960-ban végzett vegyészmérnökként a New York City College-ban, majd a Berkeley Egyetemen szerzett doktorátust.

A PhD-fokozat megszerzését követően 1963 és 1968 között a Fairchild Semiconductor kutatás-fejlesztési részlegénél dolgozott.

1969-ben, Gordon Moore társaként megalapította az Intel Corporationt, amely ma a világ legnagyobb félvezetőgyártó vállalata. 1979-től a cég elnöke, 1987-től vezérigazgatója, 1997-től pedig a vállalat vezérigazgatója és igazgatótanácsának elnöke egy személyben. A vezérigazgatói posztról 1998-ban lemondott, de 2005 májusáig az igazgatótanács elnöke maradt. Azóta rangidős tanácsadó az Intel munkájában.

A Neumann-féle számítógépben a mikroprocesszor számol, az gondolkozik, mialatt a számítógép többi szerve tárolja, megjeleníti, akár ki is nyomtatja az információt.

Ötven évvel ezelőtt az ENIAC-ban 17000 vákuum-elektroncső dolgozott, de a tranzisztor alkalmazása forradalmasította az elektronikai lehetőségeket.

1982-ben az INTEL által gyártott „286” jelzésű mikroprocesszorban már 130000 tranzisztor dolgozott, 130000 elektroncsővel egyenértékű módon, de sokkal gyorsabban. Három évvel később 1985 októberében a „386” jelzésű mikroprocesszorban 275000 tranzisztor volt. További 3 év múlva (1989 áprilisában) a „486” jelű mikroprocesszorban 1200000 tranzisztor számolt. Három év elteltével 1993 márciusában jött ki a piacra az INTEL 586 mikroprocesszor, becenevén PENTIUM, ami 3100000 tranzisztor munkáját végzi el. A következő 686 processzor-lapkában 1995. november-, aminek beceneve PENTIUMPRO, már 5500000 tranzisztor dolgozik. A fejlesztés egyre gyorsult, a 20. század 10 millió tranzisztorttal ekvivalens mikroprocesszorral zárt. Az INTEL vállalat (értsd: INTElligens ELEktronics) kezében van a világ mikroprocesszor-piacának 75%-a, évi forgalma meghaladja a 10 milliárd dollárt.

A Kék Óriás, az IBM - INTEL mikroprocesszort és MICROSOFT operációs rendszert használ.

3.5 Simonyi Károly (Charles Simonyi)

(Budapest, 1948. szeptember 10. –) szoftverfejlesztő, a szándékorientált programozás (Intentional Programming, IP) kutatója.

Második magyarként kétszer is járt a világűrben. Ő volt az ISS eddigi ötödik és hetedik űrturista-látogatója.



A számítástechnikával először középiskolásként került kapcsolatba, amikor éjjeliőrként egy szovjet Ural-2 típusú elektroncsöves számítógép vezérlőtermére vigyázott. Az egyik mérnök megtanította a gép programozására, és az ifjú Károly 18 éves korában már fordítóprogramokat készített, sőt egyik programját egy állami vállalat megvásárolta. 1968-ban az Amerikai Egyesült Államokba költözött.

1981 októberében, amikor a MICROSOFT-nak még csak 32 alkalmazottja volt, a cég munkatársa lett. (Ma a MICROSOFT-nak több tízezer alkalmazottja van.)

A korai 1980-as években APPLE-MICROSOFT együttműködésben, Steve Jobs, Bill Gates és Charles Simonyi munkája nyomán megszületett a barátságos MACINTOSH számítógép színes grafikával és egérrel.

Simonyi Károly első nevezetes alkotása volt a MULTIPLAN táblázatszerkesztő (spreadsheet), menüvel indítva. BRAVO-tapasztalataira támaszkodva 1981-ben megkezdte a WORD szövegszerkesztő kidolgozását egéralkalmazásával, ami már többféle betűtípust ajánl, és munka közben lehetőséget kínál a szerkesztés alatt álló szöveg tördelésének megtekintésére.

A MICROSOFT elnöke, Bill Gates kijelentette: „Meg fogjuk teremteni a világ legszebb táblázatszerkesztőjét!”, Simonyi Károly és Jabe Blumental valóban létrehívta az EXCEL-t.

Simonyi Károly vezette be a programozásba a "magyar stílusú elnevezést": az egyes adatcsomagok elnevezésére nem rövid és értelmetlen betűszavakat ajánlott, mint például blabla, nem is hosszú magyarázkodó nevet, hanem a név első része az adattípust, második része az adat jelentését adja. Peter Norton melegen ajánlja ezt a "magyar stílust".

Simonyi Károly és Scott McGregor megalkotta a „Windows” (ablakok) operációs rendszert: "ablakokat" tárt ki, hogy bepillanthassunk a számítógép agyába. A WINDOWS-rendszer előnye, hogy csatlakoztatható a világ minden számítógéptípusára, függetlenül a gépet gyártó vállalattól. Ma a személyi számítógépek 90%-a MICROSOFT operációs rendszert használ.

Simonyi Károly a MICROSOFT fő rendszerépítészé lett, aki arra tanította az embereket, hogy a számítógép nem csak egy szám-végeredmény elérésére és szövegszerkesztésre szolgálhat, hanem a kölcsönhatásra alapozott grafikus rendszerével virtuális valóságot teremthet.

2002-től többedmagával saját vállalkozást indított, és már nem alkalmazottja a MICROSOFT-nak.

3.6 Szentiványi Tibor

(1931 – 2009) okleveles elektromérnök, találmánya a nagy floppy.

Szentiványi Tibor több hazai és nemzetközi szakmai szervezet tagja, - a Kiss Áron Magyar Játék Társaság, alapító elnöke, a Pro Ludo-díj alapítója és adományozója.

Igazi polihisztor volt, mert a játékokon kívül sokféle terület szakértője volt: informatika, pénz (kultúr- és technikatörténet), de tartott előadásokat, publikált sok más területen is.



3.7 Jánosi Marcell



(Budapest, 1931. december 5. – 2011. július) gépészmérnök, konstruktőr.

A mikro floppy feltalálója. Az MCD típusú hajlékonylemezt a Budapesti Rádiótechnikai Gyárban (BRG) fejlesztette ki. A BRG-ben részt vett az első modern, háromsebességes orsós magnó, a Calypso megtervezésében, nem is beszélve az akár winchesterként is használható



magnetofonmechanikáról.

A 3,5"-es méretű hajlékonylemez rendszert - 3 hüvelykes kivitelben - Jánosi Marcell dolgozta ki a Budapesti Rádiótechnikai Gyárban 1973-ban. A lemezt és a hozzá tartozó BRG MCD-1 típusjelű meghajtót szabadalmaztatták. Azonban később a Jánosi-féle floppy maradt technikatörténeti kuriózum - a nemzetközi szabadalmat az állam nem újíttotta meg, így az elveszett!

A 3 hüvelykes floppy-t 1982-től a japánok (többek között a Hitachi és a Matsushita) gyártották, és az Amdek nevű cég forgalmazta, elsősorban az Apple II-ben. A japánok bevallottan "Mr. Jánosi" találmányát felhasználva vitték végig a floppy ügyét.

Jánosi Marcell a magyar hardveripar Rubik Ernője volt: ugyanabban az évben, 1974-ben, amikor Rubik előállt a bűvös kockával, Jánosi feltalálta a 3 hüvelykes floppy lemezt, és ezzel örökre beírta magát a számítógépes perifériák történelmébe.

3.8 Dr. Náray Zsolt

(1927–1995) a Számítástechnikai Koordinációs Intézet igazgatója.



1985-től Számítástechnikai Kutató Intézet és Innovációs Központ – **SZKI** alapító főigazgatója.

Dr. Náray Zsolt a világsikernek számító OCR programok, az Optikai Karakterfelismerő programok, a „Recognita” és a „MProlog” rendszer megalkotója.

4 Tantárgyi információk

Kötelező tantárgy

TANTÁRGY ADATLAP

és tantárgykövetelmények

2013.szeptember

SZÁMÍTÁSTECHNIKA



Tantárgy kódja	Szemeszter	Követelmény	Kredit	Nyelv	Tárgyfélév
BMEVESAA103	1	0+2+0 f	2	magyar	1/1

A tantárgyfelelős személy és tanszék:

Dr. Simon András, Szervetlen és Analitikai Kémia Tanszék

A tantárgy előadója:

Dr. Simon András, egyetemi adjunktus

A tantárgy célkitűzése:

Algoritmizálási készség fejlesztése egy programnyelv elsajátításán keresztül. Alapvető mérnöki számításon elvégzését, eredmények megjelenítését segítő szoftverek (pl. táblázatkezelő) készségszintű használata.

A tantárgy részletes tematikája:

EXCEL - A táblázatkezelés alapszabályai. Cellahivatkozások, cellanevek. Adattípusok, adatmozgatás, formázás. Számolás cellákkal, függvények alkalmazása. Adatok ábrázolása. Adatsorra függvény illesztése ("trendvonal").

VISUAL BASIC FOR EXCEL - A soros programozás alapjai. Változó fogalma, típusok, kifejezések, értékadás, feltételes és ciklusszerkezetek, adatbeolvasás, -kiírás. Blokkdiagram. Szintaxis diagram. Tömb, rekord, keresés, rendezés. Műveletek egy- és kétindexes tömbökkel. Változók hatásköre, lokális és globális változók. Eljárás- és függvény deklaráció, paraméterátadás. Makrók rögzítése és átalakítása Szövegfájlok használata.

5 VBA programozási ismeretek

5.1 Gondolatok a programozásról

A számítógép-program célja mindig egy felhasználó számára hasznos feladat elvégzése. A végrehajtandó feladatok (alkalmazások, applications) lehetnek egyszerűbbek, de sok számolni valót tartalmazóak, vagy ugyanazt a rutint kell sokszor elvégezni. Érdemes programot írni akkor is, ha bonyolult, összetett számítások elvégzése és sorozatos kiírása a feladat!

Minden feladat megoldásakor óhatatlanul felvetődik a kérdés, hogy érdemes-e erre a célra saját programot írni? Léteznek ugyanis a legkülönbözőbb célú, „testre szabható” kész programok, ezért a piacon érdemes jól körülnézni. Egy új program kifejlesztése jóval drágább lehet, mint egy „testre szabható” program megvásárlása, de ha ezzel a megoldással nem lehet a „felhasználó” által elvárt követelményeket maradék nélkül megvalósítani, akkor mégis az egyedi program mellett kell dönteni.

Miért pont a Visual Basic (VBA) programot választottuk oktatásra?

Az 1990-es években sokan már temették a BASIC programnyelvet, mert a C/C++ és a PASCAL nyelvek átvették a piac nagy részét. A Microsoft ekkor kiadta a VISUAL BASIC-et, ami a QuickBASIC továbbfejlesztett változataként funkcionált. Eseményvezérelt nyelvjárással és az objektumorientált programozással továbbra is fenn tudott maradni ezen a kemény piacon, hasznos és egyszerű fejlesztőeszközzé vált Windows környezetben. Továbbsegítette a nyelv terjedését két variánsa: Visual Basic for Applications (VBA) az Office programcsomag makró nyelvén, a Visual Basic Script a Windows operációs rendszer scriptnyelvén vált.

2002-ben újabb ráncfelvarráson esett át és megjelent a napjaink legnépszerűbb keretrendszere a .NET. A Visual Basic .NET már teljes mértékben objektumorientált volt és csak a nevében maradt meg a BASIC szó. Szinte semmiben sem hasonlított az eredeti BASIC programhoz, de a fejlődés csak ilyen áron volt elérhető.

A régen sokak által lenézett, de végtelen egyszerűsége miatt a TV-komputerek révén elterjedt és megkedvelt BASIC programnyelv, a Microsoft tudatos stratégiája eredményeként - megtartva elődjének egyszerűségét és áttekinthetőségét - professzionális feladatok megoldására képes fejlesztőrendszerre vált. Készítői, egy sallangmentes programozást lehetővé tevő fejlesztői környezetet alkottak.

Visual Basic for Applications (VBA) tehát minden WINDOWS Office programcsomag makró nyelvén, így minden Hallgatónak egyöntetűen rendelkezésére áll az egyetemi oktatás alatt és az otthoni számítógépen is.

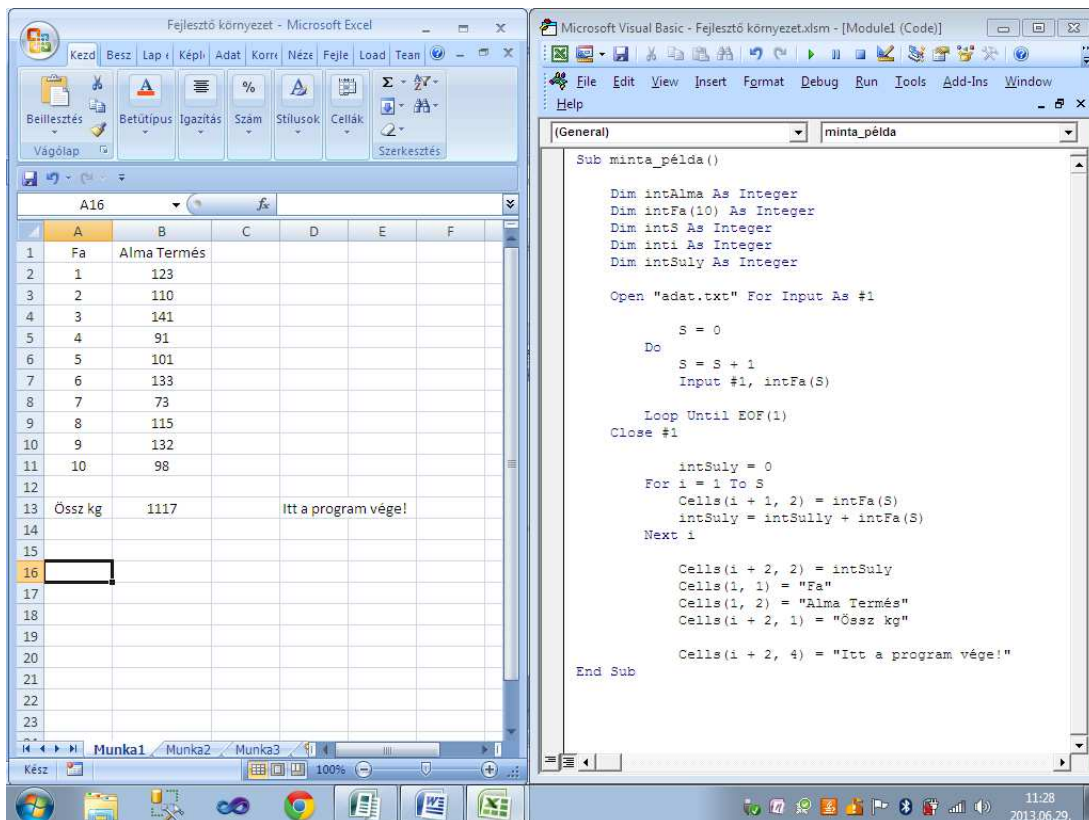
5.2 Visual Basic for Applications (VBA) fejlesztő környezete

Abból a célból, hogy a programfejlesztő felhasználónak csak magára a megoldandó problémára kelljen koncentrálni, a programgyártó cégek egyre több szolgáltatást nyújtó integrált fejlesztőrendszert adnak.

Az integrált fejlesztőrendszer azt jelenti, hogy egy erős szövegszerkesztő mögé, beépítettek olyan szükséges programokat, melyek a programíráshoz, fordításához, teszteléséhez, futtatáshoz, programmentéshez, dokumentáláshoz szükségesek. Ezen programok, a szövegszerkesztő mögött integráltan működnek együtt, a programozó minimális beavatkozása mellett.

A Windows (Ablakok) operációs rendszer lehetővé teszi, a kétablakos programozást.

A baloldali ablak az **Excel**, a jobboldali ablak a **fejlesztő környezet**.



A jobboldali ablakban látható, hogy a program a „Sub minta_pelda()” sorral kezdődik és az „End Sub” sorral végződik. A programban használt azonosítók deklarálása a „Dim” utasítással kezdődő sorokban történik, ezt követi a program parancsokat tartalmazó része, melynek futásakor tölti ki a program a baloldali Excel munkafüzetet. Azonosítót deklarálni nem csak a programon belül lehet, hanem a „Sub minta_pelda()” sor előtt is, az így deklarált változók nemcsak a „minta_pelda” programban, hanem a többi, a fejlesztői környezetben a „minta_pelda” program alá írt programokban is használhatóak.

A programot alaposan szemlélve látható, hogy a program áttekinthetősége egyrészt szavak kék színű kiemelésének, másrészt a program szövegének blokkosításával (például a „Dim” utasítások egymás alatt vannak, nem össze-vissza a szövegben) és a leírt szavak balról történő, különböző mértékű behúzásának köszönhető.

5.3 A VBA fejlesztőrendszer fogalmai

Forrásnyelvű programleírás:

A VBA szabályai szerint a sorokba írt utasítássorozat alkotja programleírást. Az ember számára olvasható (alfa-numerikus) program leírás az alapja – „forrása” az automatikusan működő fordítóprogramnak (compiler).

Fordítóprogram:

A fordítóprogram feladata, hogy az „olvasható és értelmes” nyelven írt forrásprogramot, a számítógép számára „érthető” kettes-számrendszerű kódsorozattá alakítsa át.

Ez a fordítóprogram automatikusan, a programozó közreműködése nélkül végzi a feladatát.

(... a VB fordítója „nem natív” (a gép által közvetlenül végrehajtható) utasításokká kódolta a forrásnyelvű szöveget, hanem egy közbenső nyelvre (p-kódra). A program végrehajtásakor a VB futtató magja e p-nyelvű programot interpretálja. Ennek az interpreternek feladata, hogy a p-nyelvű utasításokat natív-kódú utasításokká fordítsa, majd azokat a processzorral végrehajtsa. E program-végrehajtási módot, a TV-komputerek kicsi memória és tárolókapacitása kényszerítette ki, de annyira általánosan alkalmazták, hogy a p-nyelv egyfajta gép-független standarddává vált.)

Debugging:

A VBA fejlesztőrendszer, a hibafelderítéséhez egy hatékony, ún. Debugging (hibajavító) eszközt használ. A program tesztelése annak ellenőrzésére szolgál, hogy a program a tervezettnek megfelelően működik-e? A tesztelés többszintű, többfeladatú tevékenység, melyet ráadásul a programírás különböző fázisaiban kell végezni. A teszteléshez szorosan kapcsolódó tevékenység a feltárt hibás működések okainak a felderítése is:

- a **szintaktikai hibák** felderítése, nyelv szabályainak be nem tartásából eredő vizsgálat, már parancssoronként aktivizálódik,
- értelmezhető jelzéseket ad a hibákról, de összetettebb nyelvi struktúrák hibafelismerésére nem mindig képesek,
- a **szemantikai hibák** felderítése az algoritmus elvi hibáira, az algoritmus nem a célnak megfelelő működésére ad jelzéseket,
- a **futás közbeni hibák** felderítése is folyamatos, s ha van találat, akkor arra utaló kijelzéseket ad,
- egyes hibák olyan jellegűek, melyek végrehajtására a program nincs felkészítve, ezért ezek a program azonnali befejezését eredményezik, s ezek az ún. **run-time** (futásidejű) hibák,
- más hibák viszont ugyan nem állítják le a programfutást, csak éppen a program nem a kívánt módon működik, ezek a **program-logika** hibák.

5.4 A Visual Basic programnyelve

A működő programok tevékenységeinek tárgyai mindig valamilyen objektumok, melyek lehetnek olyan elemiek, mint a gép memóriájának egy bitje, bájtja (byte), de lehetnek ezekből felépített, egyszerűbb vagy bonyolultabb struktúrák (számok, szövegek, komplex adatok ...stb.).

A programozás során az objektumok tulajdonságai, tartalmi értékei módosíthatók, ezáltal az általuk hordozott információk igény szerint változtathatók.

Általános elnevezési szabályok:

A VB programozásban létező objektumok és azonosítók (mint speciális objektumok) elnevezésére általános szabály van. A név egy betűvel kezdődő, legfeljebb 255 karakter hosszúságú szöveg, mely a második karakterétől kezdve számjegy, - vagy aláhúzás karakter (_) is, de nem lehet azonos a VB kulcsszavainak egyikével sem. A nevek egyikében sem használatosak:

- a magyar ékezetes betűk, (még akkor sem, ha a Windows magyar nyelvű operációs rendszere lehetővé tenné),
- VB kulcsszavak (VB parancsok, illetve VB belső függvények nevei),
- a változók adattípusára jellemző karakterek, szóköz és pont.

Az azonosítók használata, láthatósága és élettartama:

Az azonosítók használatát elsősorban a láthatósága és az élettartalma határozza meg. E tulajdonságokat az ún. deklarációs utasítás jelöli meg. Az egyes azonosítókra a programnak csak meghatározott részein lehet hivatkozni (csak ott használhatók), ezen részek összességét az azonosító **hatáskörének** nevezik (lásd a „Visual Basic for Applications (VBA) fejlesztő környezete” fejezetben leírtakat). A hatáskör lehet:

- **globális**, az azonosító a program összes moduljának (a modulokban történik a programok írása) összes alprogramjában (eljárás, függvény) látható. Deklarációs kulcsszava, utasítás neve: **Public**,
- **modulszintű**, az azonosító csak a deklarációs utasítást tartalmazó moduljának összes eljárásában látható. Deklarációs kulcsszava, utasítás neve: **Private**, **Dim**,
- **eljárásszintű**, az azonosító kizárólag a deklarációs utasítást tartalmazó eljárásában látható. Deklarációs kulcsszava, utasítás neve: **Dim**.

Az azonosító élettartama a program végrehajtásának azon periódusa, melyben az azonosító létezik, jól definiálható értékkel rendelkezik:

- **statikus**, az azonosító a program végrehajtásának teljes időszaka alatt létezik, ilyen azonosítókat deklaráló utasítás neve: **Static**,
- **lokális**, az azonosító csak abban az eljárásban rendelkezik jól definiált értékkel, melyben látható is. Ebből kilépve azonban az azonosító értéke definiálatlanná válik.

5.5 Változók és konstansok használata

A változók és a konstansok a program legegyszerűbb objektumai, mert csak egy meghatározott típusú adat (szám, szöveg, logikai érték) tárolására képesek. Ezen információn kívül nincs egyéb tulajdonságuk és önálló cselekvéseik (szaknyelvre fordítva metódusok) sem.

A változó olyan azonosító, amely az információ ideiglenes tárolására szolgál, s használatának bevezetését a változó-deklarációs utasítás adja meg.

A változó-deklarációs utasítás feladata a kulcsszavak által meghatározott élettartamú és hatáskörű, adattípusú azonosítók megnevezése, melyekhez a nyelv fordítóprogramja az élettartamukra rendelt memóriaterületen, az azonosítók típusának megfelelő információt tárol.

Az alapvető adattípusú változók deklarálásának szintaxisa:

```
{Public ;Private; Dim} <változó-név> As <adattípus>
```

A <változó-név> írására az általános elnevezési szabályok érvényesek.

A <változó-név> kialakítására a Simonyi féle "magyar stílusú elnevezés" irányadó.

Az <Adattípus> kiválasztása az alábbi alapvető adattípusok alapján történhet (lásd az „Alapvető adattípusok” fejezetet)

Példák:

```
Dim intAdat_1 As Integer  
Dim lngAdat_2 As Long  
Dim sngSzam1 As Single, dblSzam2 As Double  
Dim strSzoveg As String
```

A tömbváltozók feladata a program által azonos kezelésmódot igénylő, azonos adattípusú nagyszámú változó egy lépésben történő deklarációja úgy, hogy mind az egyedi, mint az egy tömegben történő használatának lehetősége biztosítva legyen. Ennek érdekében a tömbváltozók deklarálása kiegészül azzal, hogy a <változó-név> után irt kerek zárójelpárba „()” - <változó-név>(m,n) - meg kell adni a változótömb méretét. Az egydimenziós változótömb esetében a zárójelpár egy értéket kell megadni, míg a kétdimenziós változótömb esetében a zárójelpárba, vesszővel elválasztott két értéket kell írni. „Egydimenziós” be-mutató példák:

```
Dim intAdat_1(7) As Integer  
Dim lngAdat_2(5) As Long  
Dim sngSzam1(5) As Single, dblSzam2(10) As Double  
Dim strSzoveg(35) As String
```

„Kétdimenziós” bemutató példák:

```
Dim intAdat_1(7,7) As Integer
Dim lngAdat_2(5,2) As Long
Dim sngSzam1(5,5) As Single, dblSzam2(10,4) As Double
Dim strSzoveg(35,2) As String
```

Ha a dimenzionálásnál még nem határozható meg a tömb (dinamikus tömb) mérete, akkor azt üres zárójel-párral kell megadni. „Egydimenziós” dinamikus tömb deklarálását bemutató példák:

```
Dim intAdat_1() As Integer, lngAdat_2() As Long
```

Később a program futása során, még a változó használata előtt, a dinamikusnak deklarált változók aktuális méretét ReDim, vagy ReDim Preserve parancsok után, határozottá kell deklarálni.

```
ReDim intAdat_1(6) As Integer
intAdat_1(6) = 22
```

A ReDim parancssor újradimenzionálja intAdat_1 változó méretét úgy, hogy a tömb minden elemének értékét törli, majd a tömb 6. helyére történik a „22” tárolása.

```
ReDim Preserve lngAdat_2(10) As Long
```

A ReDim Preserve parancssor újradimenzionálja intAdat_1(10) változót úgy, hogy a 10. memória előtti elemeket nem nullázza, s a tárolás a 10. elemre fog történni.

A konstans olyan azonosító, amely az információ ideiglenes tárolására szolgál, s használatának bevezetését a változó-deklarációs utasítás adja meg. Értéke a program során NEM változtatható.

konstans adattípusú azonosító deklarálásának szintaxisa:

```
{Const} <konstans-név> As <adat-típus> = <érték>
```

Példák:

```
Const intAdat_1 As integer = 5
Const szoveg1 as string = „van”, szoveg2 as string = "nincs"
```


5.6 Alapvető adattípusok

Az adattípus az állandók és a változók által hordozott információ jellegét és lehetséges értéktartományát határozza meg. A adattípusok az információ jellege alapján lehet: numerikus, szöveges, logikai, dátum-idő csoportokba sorolhatók. Fontosabb numerikus adattípusok:

- **yte**: 1 bájtól tárol egy $[0, 255]$ intervallumbeli 10-es számrendszerbeli egész számot;
- **Boolean** (logikai): 2 bájtól tárol egy logikai értéket (a False a csupa 0 bitekből álló-, a True pedig a legalább egy bitjén nem 0 értékű duplabájtnak felel meg);
- **Integer** (egész): 2 bájtól tárol egy $[-32768 \leq x \leq 32767]$ intervallumbeli 10-es számrendszerbeli egész számot;
- **Long** (egész): 4 bájtól tárol egy $[-2147483648 \leq x \leq 2147483647]$ intervallumbeli 10-es számrendszerbeli egész számot (kódolás, stb. mint előbb);
- **Single** (-precision floating-point; egyszeres pontosságú lebegőpontos): az IEEE 32-bites szabvány szerint tárolja az $1.401298 \cdot 10^{-45} \leq x$ abszolút értéke $\leq 3.402823 \cdot 10^{38}$ feltételnek eleget tevő 10-es számrendszerbeli valós számot (lassú lebegőpontos aritmetika);
- **Double** (dupla pontosságú lebegőpontos): az IEEE 64-bites szabványa szerint tárolja a $4.94065645841247 \cdot 10^{-324} \leq x$ abszolút értéke $\leq 1.79769313486232 \cdot 10^{308}$ feltételnek eleget tevő 10-es számrendszerbeli valós számot (még lassúbb aritmetikájú);
- **Currency** (- pénz): 8 bájtól, skálázott egész számként kódolva tárol egy $-922337203685477.5808 \leq x \leq +922337203685477.5807$ intervallumbeli 10-es számrendszerbeli valós számot (gyors bináris aritmetika);
- **Decimal** (decimális): 12 bájtól tárol egy kb. 28 jegyű 10-es számrendszerbeli előjeltelen egész számot, ami a $[0, 1028]$ határon belül változtathatóan skálázható;
- **String** (szöveges): egy 10 bájtól adatfejet követően a szöveg karakterei állnak, a szöveg maximális hossza kb. 2^{31} karakterben van limitálva; kicsit lassúbb kezelést biztosít;
- **Variant**: a variant adattípus tárolhat mind számot, mind szöveget, de memóriaigénye jóval nagyobb (szám: 16 bájt, szöveg: 22 bájtól adatfejet + szöveg hossza)

Az alapvető és gyakrabban használt adattípusokat különböző jelekkel is meg lehet adni:

Adattípus:	Integer	Long	Single	Double	String
Jel:	%	&	!	#	\$

Természetesen az adattípusok jelei a változók deklarálásánál is használhatóak:

```
Dim intAdat_1%
Dim lngAdat_2&(5)
Dim sngSzam1!(5,5), dblSzam2#(10,4)
Dim strSzoveg$(35,2)
```

5.7 Aritmetikai, relációs, logikai konverziós és szövegkezelő műveletek, matematikai függvények

Aritmetikai műveletek:

- összeadás, jele: +,
- kivonás, jele: -,
- szorzás, jele: *,
- osztás, jele: /,
- egészszámosztás: a nem egész számok előzőleg egészre kerekítődnek, az osztás eredményének tört része elvész, jele: \,
- maradékképzés: a nem egész számok előzőleg egészre kerekítődnek, az osztás eredményének egész része elvész, parancsa: Mod,
- hatványozás, jele: ^, a ^ előtti szám a hatvány-alap, a ^ utáni szám a kitevő,

Relációs műveletek:

- egyenlőség, jele: =,
- egyenlőtlenség, jele: <> ,
- kisebb, jele: < ,
- nagyobb, jele: > ,
- kisebb vagy egyenlő, jele: <= ,
- nagyobb vagy egyenlő, jele: >= .

Szövegkezelő művelet:

A szövegek, illetve a szöveg és a nem szöveges adat összefűzésére szolgál (a nem szöveges adat előzőleg szövegessé konvertálódik, jele: &.

Matematikai függvények:

- Abs(<szám>): a <szám> abszolút értékét adja meg,
- Sin(<radián>): a szinusz értéket számítja ki,
- Cos(<radián>): a koszinusz értéket számítja ki,
- Tan(<radián>): a tangens értéket számítja ki,
- Atn(<radián>): az arc tg értéket számítja ki, π értéke $4 * \text{atn}(1)$ értékével egyenlő,
- Round (<szám>[,<tizedes>]) egészre, vagy adott tizedesre kerekít.
- Fix (<szám>): <szám> egész értékre kerekítve adja meg: Fix(-4.3) értéke -4, Fix(4.3) értéke 4, Round (<szám>,0) parancsnak felel meg,

- Int(<szám>): <szám> egészrészét adja meg: Int(-4.3) értéke -5, Int(4.3) értéke 4,
- Exp(<szám>): az $e^{<szám>}$ ($e = 2,718282$) értékét adja meg,
- Log(<pozitív>): a <pozitív> szám természetes logaritmusát (ln) adja meg,
- Sqr(<nem-negatív>): a <nem-negatív> szám négyzetgyökét adja meg,
- Rnd: véletlen szám generálása 0 és 1 közötti értékben,

Konverziós függvények:

- Str(<num-kifejezés>): a megfelelő decimális számot szövegesen jeleníti meg,
- Val(<szöveg>): számot tartalmazó <szöveg> szám-értékét adja meg,
- ASC(<szöveg>): <szöveg> első karakterének ANSI-kódja,
- Chr(<ANSI-kód>): <ANSI-kód> értékének megfelelő karakter,
- Hex(<egész>): az <egész> hexadecimális formáját jeleníti meg,
- Okt(<egész>): az <egész> oktális formáját jeleníti meg,
- CStr(<változó>): a <változó>-t szöveggé alakítja át,
- CSng(<változó>): a <változó>-t single típusú változóra alakítja át (amennyiben lehetséges),
- CDbl(<változó>): a <változó>-t double típusú változóra alakítja át (amennyiben lehetséges),
- CInt(<változó>): a <változó>-t integer típusú változóra alakítja át (amennyiben lehetséges),
- CLng(<változó>): a <változó>-t long típusú változóra alakítja át (amennyiben lehetséges),
- CByte(<változó>): a <változó>-t byte típusú változóra alakítja át (amennyiben lehetséges),
- CCur(<változó>): a <változó>-t currency típusú változóra alakítja át (amennyiben lehetséges),
- CDec(<változó>): a <változó>-t decimal típusú változóra alakítja át (amennyiben lehetséges),
- CBool(<változó>): a <változó>-t boolean típusú változóra alakítja át (amennyiben lehetséges),
- CVar(<változó>): a <változó>-t variant típusú változóra alakítja át.

Mintapéldák:

Kifejezés	Érték	Megjegyzés
Chr(100)	d	
Asc("d")	100	
Val("1111, Bp. Szt. Gellért tér 4.")	1111	
CInt(56.2)	56	
CInt(100000)	„Overflow”	hibaüzenet
CStr(56.2)	56,2	szöveg
Hex(500)	1F4	
Oct(500)	764	

Logikai műveletek:

- And: és: feltételek összefűzésére szolgál, az eredmény csak akkor igaz, ha mindkét feltétel teljesül. Például " $x \geq 3$ And $x \leq 7$ " eredménye akkor igaz, ha „x” értéke 3 és 7 közé esik.
- Or: vagy: feltételek összefűzésére szolgál, az eredmény csak akkor igaz, ha bármelyik feltétel teljesül. Például " $x < 3$ Or $x > 7$ " eredménye akkor igaz, ha „x” értéke 3-nál kisebb vagy 7-nél nagyobb, míg a " $x \geq 3$ OR $x \leq 7$ " eredménye mindig igaz.
- Xor: kizár: feltételek összefűzésére szolgál, az eredmény csak akkor igaz, ha a feltételek között van olyan, amely teljesül és van olyan amelyik nem. Például " $x \geq 3$ Xor $x \leq 7$ " eredménye akkor igaz, ha „x” értéke 3-nál kisebb vagy 7-nél nagyobb.
- Eqv: azonos: feltételek összefűzésére szolgál, az eredmény csak akkor igaz, ha az összes feltétel igaz vagy hamis. Például az " $x < 3$ Eqv $x > 7$ " eredménye akkor igaz, ha „x” értéke 3 és 7 közé esik.
- Not: tagad: logikai változó értékét változtatja az ellentétére, ha „x” értéke „true”, akkor Not x értéke „false” lesz.

5.8 A Visual Basic utasításai és parancsai

A Visual Basic utasítások jellemzője az átláthatóság, a funkcionálisan azonos parancsok elkülönülése, a „jobbra tartó írásmód”, amihez az üres sorok beszúrása és Tabulátor „erőteljes” használata ad lehetőséget. Az alábbi irányelveket javasolt betartani:

- Sub End Sub oszlopába semmit nem szabad írni.
- Minden parancsot lehetőleg külön sorba kell írni. Amennyiben mégis több parancsot írunk ugyanabba a sorba, akkor a parancsokat kettősponttal válasszuk el egymástól.
- Összetett parancsok kulcs sorainak kezdő oszlopába pl.(For ...Next) (If... Else ...End If) (Do ...Loop) nem szabad semmit írni, a közéjük kerülő értékadó utasításokat egy Tab-bal jobbra kell írni.
- Egymásba ágyazott, összetett parancsok írására is ugyan ez vonatkozik, (például egymásba írt két For ...Next). Ilyenkor a belső parancs egy Tab-bal már jobbra írt, majd a bennük levő értékadó utasítások újabb Tab-bal jobbra íródjanak.

5.8.1 Az „Option ...” utasítások

Az „Option” szóval kezdődő utasításokat a program írására használt modul tetején az első program kezdő „Sub” utasítása előtt írjuk be, így az egész modulra érvényesek lesznek. Az alábbi utasítások használhatóak a programok írása során:

- **Option Explicit:** az Option Explicit utasítás használata esetén csak olyan változókat használhat a program, amelyeket az „Alapvető adattípusok” alfejezetben leírtaknak megfelelően a programozó

deklarált a változó használata előtt. Enélkül a parancs nélkül a Visual Basic megengedi, hogy a változót (variant adattípussal) a program automatikusan hozza létre a változó első alkalmazása előtt (implicit deklaráció). Az Option Explicit használatával elkerülhető hogy egy változónév elírásával akaratunkon kívül új változót hozzunk létre és az értéket is kapjon.

- **Option Base:** amikor a programozó „Dim Tomb(3) as integer” sorral deklarálja a „Tomb” változót, akkor annak négy eleme lesz: Tomb(0), Tomb(1), Tomb(2), Tomb(3). Az Option Base 1, vagy Option Base 2 sor beírása esetén a „Tomb” változónak három, illetve két eleme lesz (Tomb(1), Tomb(2), Tomb(3); illetve Tomb(2), Tomb(3)).
- **Option Compare Text:** az Option Compare Text alkalmazása esetén a Visual Basic nem tesz különbséget az adott karakter „kisbetűs” és „nagybetűs” változata között. String típusú változók vizsgálatának esetén használható.

5.8.2 Műveletek szöveges változóval

A szöveges adattípusok szövegek tárolására szolgál (belsőleg Unicode kódolással). A string által foglalt memória terület egy 10 bájtos adatfejet és ezt követően a szöveg karaktereit tárolja. Szöveges változón az alábbi 5 művelet hajtható végre:

- A szöveghossz vizsgálat a Len() paranccsal történik. Szintaxisa:

`<hossz_változó> = Len (<szöveg>)`

A Len() parancs argumentumába (zárójelbe) beírt szöveg vagy szövegváltozó értékének megfelelő szöveg karaktereinek számát adja vissza a <hossz_változó>-ba.

- A szöveg elejének vizsgálata a Left() paranccsal történik. Szintaxisa:

`<szöveg_változó> = Left(<szöveg>,<karakter_száma>)`

A Left() parancs argumentumába beírt szöveg vagy szövegváltozó szövegéből, a szöveg elejéről a <karakter_száma>-nak megfelelő számú karaktert ad vissza a <szöveg_változó>-ba.

- A szöveg végének vizsgálata a Right() paranccsal történik. Szintaxisa:

`<szöveg_változó> = Right(<szöveg>,<karakter_száma>)`

A Right() parancs argumentumába - zárójelbe - beírt szöveg vagy szövegváltozó szövegéből, a szöveg végéről a <karakter_száma>-nak megfelelő számú karaktert ad vissza a <szöveg_változó>-ba.

- A szöveg bármely részének vizsgálata a Mid() paranccsal történik. Szintaxisa:

`<szöveg_változó> = Mid(<szöveg>,<karakter_száma>,<utána_száma>)`

A Mid() parancs argumentumába beírt szöveg vagy szövegváltozó szövegéből, a szöveg elejétől a <karakter_száma>-nak megfelelő számú karaktertől, az <utána_száma>-nak megfelelő számú karaktert ad vissza a <szöveg_változó>-ba.

- Szövegek „összeadása” az „&” karakterrel történik. Szintaxisa:

`<szöveg_változó_teljes> = <szöveg> {& <szöveg>} {& <szöveg_változó>}`

A & karakter (karakterek) összefűzik a <szöveg>-eket, vagy <szöveg_változó>-kat egy „szöveg-törzsbe”, s ez kerül át a <szöveg_változó_teljes> változóba.

Program

Sub szoveg()

Dim strNev As String, intX%, strNevresz\$

strNev = "Vegyeszmérnöki Kar"

Cells(3, 2) = strNev

intX = Len(strNev)

Cells(3, 4) = intX

strnevresz = Left(strNev, 3)

Cells(3, 5) = strnevresz

strnevresz = Right(strNev, 3)

Cells(3, 6) = strnevresz

strnevresz = Mid(strNev, 5, 3)

Cells(3, 7) = strnevresz

strnevresz = Left(strNev, 7) & " " & Right(strNev, 3)

Cells(5, 2) = strnevresz

End Sub

Megjegyzés

	A	B	C	D	E	F	G
1		Szöveg		Hossz	Eleje	Vége	Közepe
2							
3		Vegyeszmérnöki Kar		18	Veg	Kar	ész
4							
5		Vegyesz Kar					
6							
7							

Szöveghossza - karakterszám

Szöveg eleje

Szöveg vége

Szöveg közepe

Szöveg összefűzése

- A Like parancs segítségével meg tudjuk mondani, hogy a szöveg tartalmazza vagy sem a programozó által megadott karaktersort. Ha „Eredmeny” változó boolean típusú, akkor az

Eredmeny = Opera Like "O*"

parancs után „Eredmeny” változó értéke true lesz, mert az Opera szó valóban „O” betűvel kezdődik. Az alábbi táblázat a Like parancs alkalmazására mutat néhány példát.

Összehasonlítás	Eredmény	Megjegyzés
"Motor" Like "M*"	True	A * egy vagy több karaktert helyettesít.
"Motor" Like "?o*"	True	A ? egy karaktert helyettesít.
"3-as" Like "#-as"	True	A # egy számkaraktert helyettesít
"u-as" Like "#-as"	False	
"3-as" Like "?-as"	True	
"u-as" Like "?-as"	True	
"Jani" Like "Jan[ió]"	True	A [ió] az i-ó tartományban egy karaktert helyettesít.
"Jani" Like "[A-M]*"	True	
"Jani" Like "[!A-M]*"	False	A ! kizárja az A-M tartomány karaktereit.

5.8.3 Cellaparancsok

5.8.3.1 Cella használata írásra - olvasásra

A Cella (Cells) a munkalap (WorkSheet) és a tartomány (Range) objektumnak egy olyan tulajdonsága, mely paramétereiben hivatkozott cellához a megfelelő - egyetlen cellát tartalmazó - Range típusú objektumot rendeli. Szintaxisa:

`<objektum>.Cells(<sor száma>,<oszlop száma>),`

ahol az objektum lehet, Worksheets(„Munka1”), Worksheets(1) (munkalap nevével, vagy azonosító számaival) vagy Range és a <sor száma>,<oszlop száma> a cellát azonosító egész számok.

Megjegyzés:

Ha a cella szintaxisból az objektum elmarad, akkor a cellautasítás mindig az éppen aktív munkalapra lesz érvényes, s ott hajtódik végre.

A Cella megadása lehet:

- „direkt”, a sor és oszlop számok konstans egész számok (pl. Cells (2, 5)),
- „indirekt”, a sor és oszlop számok változó egész számok (pl. Cells (x, y)),
- „összetett”, a sor és oszlop számok számolással képzett- egész számok (pl. Cells (x + 4, y + i)).

A cellautasítások, kiírások és beolvasások módját jól szemlélteti az alábbi program:

Program

Sub cella_muveletek()

```
Dim intAdat As Integer
Dim sngEredmeny As Single
Dim intX As Integer
Dim intY As Integer
```

Cells(2, 3) = 23

intX = 3

intY = 4

Cells(intX, intY) = 25

intAdat = Cells(2, 3)

sngEredmeny = Cells(3, 4) / intX

Cells(5, 5) = "a PROGRAM VÉGE"

End Sub

Megjegyzés

	A	B	C	D	E	F
1						
2			23			
3				25		
4						
5					a PROGRAM VÉGE	

Kiírás cellába
"direkt címzéssel".

Kiírás cellába
"indirekt címzéssel".

Beolvasás cellából.

Beolvasás cellából művelettel.

Ajánlott a PROGRAM VÉGE megjelenítése.

5.8.3.2 Cellák tartalmának törlése

A cella és a cellatömbök tartalmának törlésének szintaxisa:

<objektum>.ClearContents

Célszerű arról gondoskodni, hogy mindig „üres” cellákba, cellatartományokba történjen a kiírás. A programfejlesztés közben nagy a valószínűsége annak, hogy egyre több „cellában felejtett kiírás” lesz látható az Excel munkalapon. Ezért gondoskodni kell arról, hogy az előző kiírás adatai le legyenek törölve a munkalapról az új adatok kiírása előtt. A teljes (Active) munkalap (Workshhets) tartalmának törlése lehetséges az:

- az EXCEL-ben az egérrel a mindet kijelöl gombra (balfelső sarok, az „A” oszlop fejléce mellett és az „1” sor felett), majd a billentyűzetten a „Delete” gombot lenyomva,
- Programból:

a)

```
Cells.Select
Selection.ClearContents
Range("A1").Select
```

A törlésre a második parancssorban kerül sor, a harmadik parancssor csak a munkalap cellák kijelölésének megszüntetéséhez szükséges.

b)

`Worksheets("Munka1").Cells.ClearContents`
`Worksheets(1).Cells.ClearContents`

Míg az a) esetben az Excel aktív munkalapjáról tudunk adatokat törölni, addig a b) esetben bármely munkalap tartalma törölhető.

A fenti parancsok minden cellabeírást törölnek, ezért ezekkel óvatosan kell bánni!

Az oszlopok és a sorok tartalmának törlési lehetőségeit az alábbi példák mutatják meg:

Parancssor

`Columns("A").ClearContents`
`Columns(1).ClearContents`
`Rows(3).ClearContents`
`Columns("E:G").ClearContents`

Megjegyzés

az A oszlop tartalmának törlése
az A oszlop tartalmának törlése
a 3. sor tartalmának törlése
az E, F és G oszlopok tartalmának törlése

A `Worksheets("Munka2").Columns(1).ClearContents` vagy `Worksheets(5).Rows(3).ClearContents` parancsok alapján készített utasításokkal bármely munkalap bármely oszlopának - sorának tartalma törölhető.

Cellák vagy cellaterületek tartalmának törlése az alábbi módokon lehetséges:

Parancssor

`Range("A2").ClearContents`
`Cells(2, 1).ClearContents`
`Cells(2, 1).Select`
`Selection.ClearContents`
`Range("C3:E8").ClearContents`
`Range(Cells(3, 3), Cells(8, 5)).ClearContents`

Megjegyzés

A2 cella tartalmának törlése
A2 cella tartalmának törlése
A2 cella tartalmának törlése
a C3-E8 cellatömb tartalmának törlése
a C3-E8 cellatömb tartalmának törlése

5.8.4 Adatbevitel és adatkivitel objektumokkal

5.8.4.1 InputBox - A VBA program adatbeviteli objektuma

Az InputBox objektum alábbi lehetséges tulajdonságai közül 3 tulajdonságát javasolt megadni a zárójelben:

```
InputBox(Prompt, [Title], [Default], [XPos], [YPos], [HelpFile], [Context]) As String
```

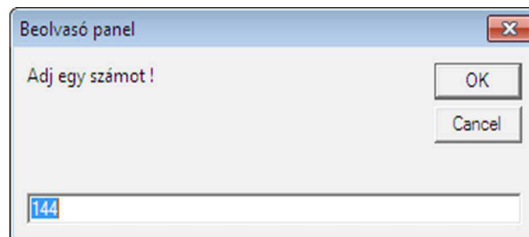
- **Prompt** (közlés): a szürke mezőben jelenik meg, információt ad a felhasználónak arról, hogy mit várunk el tőle,

- **Title** (cím): a kék fejlécben jelenik meg,
- **Default** (induló, javasolt adat) - a TextBox-ban jelenik meg, sötét alapon,

Amennyiben az

```
x = InputBox("Adj egy számot!", "Beolvasó panel", 144)
```

parancsot adjuk ki, akkor a jobboldalon lévő ablak tűnik fel.

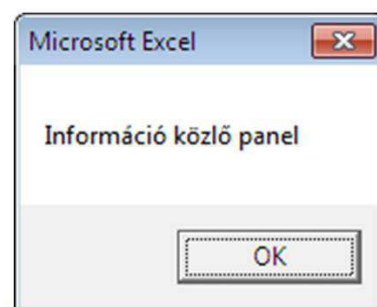


5.8.4.2 MsgBox - A VBA program adatkiviteli objektuma

Az MsgBox objektumnak minimum egy tulajdonságát, a Prompt (közlés) tulajdonságát meg kell adni, mely a felhasználó részére szolgáltat információt és a szürke mezőben jelenik meg.

```
MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile],  
[Context]) As VbMsgBoxResult
```

MsgBox "Információközlő panel"



parancs kiadása után a jobboldalon lévő ablak jelenik meg.

Amennyiben többféle információt szeretnénk kiíratni, akkor az egyes részek a „&” jellel köthetőek. Ha például az „x” változó értékét szeretnénk kiírni, akkor célszerű az alábbi parancsot kiadni:

MsgBox "x = " & x

5.8.5 Adatbevitel és adatkivitel *.txt fájl segítségével

A VBA programba text típusú fájlokból (általában *.txt fájlok) lehet adatokat beolvasni, illetve ilyen fájlokba lehet adatokat kiírni. A *.txt fájlok adatállománya az ASCII kódtábla kódjait alkalmazza. Számokon és betűkön kívül a space, tab, enter, EOF (file vége, end of file) vezérlő karaktereket ismeri fel és használja a VBA.

A VBA programban az OpenInput...Close parancsutasítás látja el a feladatot úgy, hogy az „Input” parancs kijelölt *.txt fájlból, balról - jobbra haladva „karakterláncokat” olvas be a megadott változóba. Egy „karakterláncnak” tekinti az „Input” utasítás azt a szöveg vagy számkarakter sorozatot, melyet:

- szám, illetve idéző jelek között lévő szöveg esetén beolvasása esetén vessző, pontosvessző, space, tab, enter, EOF vezérlő karakter,
- míg idézőjelek nélkül lévő szöveg esetében enter, EOF vezérlő karakter zár le.

Az Open ... Close utasítás együttes szintaktikája:

Open <fájlnev.txt> For <alkalmazás - Input, Output vagy Append> As # <virtuális memórianev>
Input #<a virtuális memória neve>, <eltárolás helye - változója>

illetve fájlba történő írás esetén:

Write #<a virtuális memória neve>, <eltárolás helye - változója>

Végezetül a fájlt be kell zárni:

Close #<a virtuális memória neve>

Az „Open” megnyitja a <fájlnev.txt> -t beolvasásra (Input), kiírásra (Output), vagy hozzáfűzésre (Append) és a beolvasás vezérlését átadja (As) a megnevezett virtuális memória résznek.

Az „Input #” a virtuális-memória vezérlése mellett kiolvassa a megnyitott text fájlból, a kért „karakterláncot” és a beolvasott karakterláncot eltárolja a kijelölt változóba.

A „Write #” a változó tartalmát kiírja a megnyitott fájlba. Ha a parancssor végén vessző vagy pontosvessző jel van, akkor a program a következő változó értékét a fájl ugyanazon sorába fogja kiírni, különben a következő változó kiírása új sorba történik. Ezek a kiírási lehetőségek jól alkalmazhatóak kétdimenziós tömbök esetén, mert így a tömb azonos soraiban lévő változók a fájlban is azonos sorba kerülhetnek, így a fájl áttekinthetőbbé válik.

A „Close # <a virtuális memória neve>” bezárja a megnyitott text fájlt.

A beolvasó Open utasítás után, legfeljebb annyi Input parancs lehet, mint amennyi „karakterláncot” tartalmaz a megnyitott fájl. Természetesen ciklusszervező utasítások beépítésével is (For ... Next, Do ... Loop), többszörözhető az Input parancsok száma.

Amennyiben a Szamsor.txt fájlt szeretnénk megnyitni, akkor az alábbi paranccsal tehető meg:

Open "Szamok.txt" For Input As #1

A program a Szamok.txt fájlt az alapértelmezett könyvtárban fogja keresni, amennyiben nem találja, hibaüzenetet ad. Az a könyvtár számít alapértelmezettnek, amelybe az Excel fájl a megnyitása után utoljára került lementésre. Amennyiben ismert, hogy a Szamok.txt fájl az Adatok könyvtárban található, akkor a fájl megnyitása az alábbiak szerint is történhet:

Open "c:\Adatok\Szamok.txt" For Input As #1

Ugyanakkor sokszor előfordulhat, hogy a felhasználónak kell megkeresnie a beolvasandó fájlt, vagy meghatározni a mentés helyét, amit az

Open Application.GetOpenFilename For Input As #1

Open Application.GetSaveAsFilename as #1

parancsok futásakor tehet meg a Windows alapértelmezett „Megnyitás” és „Mentés másként” ablakai segítségével.

Az Open - Close utasítás együttes működését mutatja az alábbi program. A jobb áttekinthetőség végett az összetartozó parancsok kettősponttal elválasztva ugyanazon sorban vannak.

Sub beolvas()

Dim intA(5) As Integer

Dim intB(5) As Integer

Range("A1:D3").ClearContents

	A	B	C	D	E
1	25	24	23	22	
2					
3	25	24	23	22	
4					

Open Application.GetOpenFilename For Input As #1

Input #1, intA(1): Cells(1, 1) = intA(1)

Input #1, intA(2): Cells(1, 2) = intA(2)

Input #1, intA(3): Cells(1, 3) = intA(3)

Input #1, intA(4): Cells(1, 4) = intA(4)

Close #1

Open "Szamok.txt" For Input As #1

Input #1, intB(1): Cells(3, 1) = intB(1)

Input #1, intB(2): Cells(3, 2) = intB(2)

Input #1, intB(3): Cells(3, 3) = intB(3)

Input #1, intB(4): Cells(3, 4) = intB(4)

Close #1

End Sub

A program az Open utasításaiban a Szamok.txt fájl adatait olvassa be, melyben csak 4 „karakterlánc” szerepel: 25, 24, 23 és 22. Az Input ezeket olvassa be sorban egy tömbváltozó elemeibe, s és utána a Cells paranccsal kiírja az Excel munkalapra.

5.8.6 Elágazó utasítások. Feltételes és feltétel nélküli utasítások

Az elágazó parancsutasítások lehetővé tesznek feltétel nélkül (ún. ugró utasítás), vagy feltételtől függő működést.

5.8.6.1 Go to utasítás

A feltétel nélküli elágazó parancsutasítás a parancssorok végrehajtásának monoton egymásutániságát képes megszakítani. Ilyen parancs a GoTo, melynek szintaxisa:

GoTo <címke>

A „címke” egy önálló parancssor „neve”, melyet kettőspont követ.

5.8.6.2 If ... then elágazó parancs

A feltételtől függően elágazó parancsutasítások a program utasításainak az írásuk sorrendjétől eltérő végrehajtását eredményezik oly módon, hogy egy logikai kifejezés aktuális értékétől függően az utasításokban leírt algoritmusban elágazásokat tesznek lehetővé. Jellemző példája az If then utasítás, melynek szintaxisa:

```
If <feltétel-1> Then
    <feltétel-1 igaz utasítások>
    ...
    ElseIf <feltétel-i> Then
        <feltétel-i igaz utasítások>
        ...
    Else
        <feltétel-utolsó hamis utasítások>
End If
```

Mintapélda:

Program

```
Sub felteteles_elagazo()
```

```
Dim intSzam As Integer
```

```
ujszam:
```

```
intSzam = InputBox("intSzam = ? (0-10)")
```

```
If intSzam = 0 Then
    Cells(11, 1) = "0 bevitel, nincs több beolvasás"
    GoTo vege
End If
```

```
If intSzam > 10 Then
    Cells(11, 1) = "10-nél nagyobb. új beolvasás"
    GoTo ujszam
End If
```

```
If intSzam > 7 Then
    Cells(intSzam, 1) = "Szám > 7"
    ElseIf intSzam > 3 Then
        Cells(intSzam, 1) = "Szám < 7 és Szam > 3"
    Else
        Cells(intSzam, 1) = "Szám < 3 és Szam > 0"
End If
```

```
GoTo ujszam
```

```
vege:
```

```
End Sub
```

Megjegyzés

	A	B	C
1	Szám < 3 és Szam > 0		
2	Szám < 3 és Szam > 0		
3			
4			
5	Szám < 7 és Szam > 3		
6			
7	Szám < 7 és Szam > 3		
8			
9			
10	Szám > 7		
11	0 bevitel, nincs több beolvasás		

Alsó érték ellenőrzése

Felső érték ellenőrzése

Érték szerinti kiírás

Ugró utasítás

Ha a <feltétel-1> = Igaz, akkor a Then kulcsszót követő utasítások hajtódnak végre, ellenkező esetben a program <feltétel-2> teljesülését vizsgálja. Ha <feltétel-2> sem teljesül, akkor a program továbblép, egészen addig, míg egy feltétel nem teljesül, vagy el nem jut a <feltétel-utolsó hamis utasítások> áigig. A feltételtől függően elágazó parancsutasításnak nem kötelező „Else” ágat tartalmaznia, ekkor a program nem tesz semmit, ha egyik feltétel sem teljesül.

5.8.6.3 Select Case ... End Select elágazó parancs

If then utasításhoz hasonló a Select Case utasítás, melynek szintaxisa:

```
Select Case <teszt-kifejezés>
  Case <kifejezés lista-i>
    <utasítások-i>
    ...
  Case <kifejezés lista-k>
    <utasítások-k>
    ...
  Case Else
    <máskülönben utasítások>
End Select
```

A programnak nem kötelező „Case Else” ágat tartalmaznia.

A <kifejezés> bármilyen numerikus vagy szöveges kifejezés. A <kifejezés lista-i> pedig <kifejezés-lista-i>, vagy <min-kifejezés lista-i> To <max-kifejezés lista-i>, vagy Is <alapvető-reláció-jeli> <kifejezés lista-i> formák valamelyikéből összeállított, egymástól vesszővel elválasztott lista.

Az utasítás kiértékeli <teszt-kifejezés>, és értékének vagy a <kifejezés> értékével való egyezése, vagy határaival adott intervallumba való esésével, vagy az „Is” után megadott relációs kapcsolat fennállásakor a megfelelő Case <utasítások> hajtódnak végre, hogy azok befejezése után az „End Select”-et követő utasítássor folytatódjék

Mintapélda:

Program

```
Sub felteteles_elagazo2()
Dim intSzam As Integer
```

```
ujszam:
intSzam = InputBox("intSzám = ? (0-10)")
```

```
Select Case intSzam
Case 0
  Cells(11, 1) = "0 bevitel, nincs több beolvasás"
  GoTo vege
Case Is > 10
  Cells(11, 1) = "10-nél nagyobb. új beolvasás"
  GoTo ujszam
Case Is > 7
  Cells(intSzam, 1) = "Szám > 7"
Case Is > 3
  Cells(intSzam, 1) = "Szám < 7 és Szam > 3"
Case Else
  Cells(intSzam, 1) = "Szám < 3 és Szam > 0"
End Select
```

```
GoTo ujszam
```

```
vege:
End Sub
```

Megjegyzés

	A	B	C
1	Szám < 3 és Szam > 0		
2	Szám < 3 és Szam > 0		
3			
4			
5	Szám < 7 és Szam > 3		
6			
7	Szám < 7 és Szam > 3		
8			
9			
10	Szám > 7		
11	0 bevitel, nincs több beolvasás		

Alsó érték ellenőrzése

Felső érték ellenőrzése

Érték szerinti kiírás

Ugró utasítás

5.8.7 Ciklusszervező utasítások

A ciklusszervező utasítások olyan utasítások, melyek az általuk „keretbe foglalt” utasítások csomagjának (az ún. „ciklusmagnak”) ismételt végrehajtását eredményezik. A ciklus végrehajtása lehet:

- fix számú lépésből álló ciklus : **For ... Next** ciklus
- feltételtől függő, kötetlen számú lépésből álló ciklus: **Do ... Loop** ciklus

5.8.7.1 For ... Next ciklus

A For ... Next ciklusutasítás a ciklusmagot ismétli egy numerikus típusú változónak a kezdeti értékétől a végső értékéig a <lépésköz> által definiált változás mellett. A For ... Next parancs szintaxisa:

```
For <ciklusváltozó> = <kezdeti érték> To <végső érték> Step <lépésköz>
    <utasítások>
Next <ciklusváltozó>
```

A <kezdeti érték>, a <végső érték>, valamint a <lépésköz> lehet pozitív/negatív konstans vagy változó, egész vagy tört kifejezés is. Pozitív <lépésköz> esetén a <végső értéknek>-nek, míg értelemszerűen negatív <lépésköz> esetén a <kezdeti érték>-nek kell a nagyobbbnak lenni. A Step <lépésköz> elhagyható, ebben az esetben +1 lesz az érvényes lépésköz.

Mintapéldák a For...Next ciklus alkalmazására:

Program

```
Sub for_next_1()
```

```
Dim intX As Integer, intY%, intZ%
```

```
intZ = 0
```

```
For intX = 1 To 7 Step 0.5
```

```
    intZ = intZ + 1
```

```
    Cells(intZ + 1, 1) = intX
```

```
Next intX
```

```
For intY = 2 To 14
```

```
    Cells(intY, 3) = intY
```

```
Next intY
```

```
End Sub
```

```
Sub for_next_2()
```

```
Dim intX As Integer, intY%, intZ%
```

```
For intX = 2 To 6
```

```
    For intY = 2 To 14
```

```
        intZ = intX + intY
```

```
        Cells(intY, intX) = intZ
```

```
    Next intY
```

```
Next intX
```

```
End Sub
```

Megjegyzés

Egyszerű For ... Next ciklusok

A2:A14 cellák kitöltése.

C2:C14 cellák kitöltése.

	A	B	C
1			
2	1		2
3	1,5		3
4	2		4
5	2,5		5
6	3		6
7	3,5		7
8	4		8
9	4,5		9
10	5		10
11	5,5		11
12	6		12
13	6,5		13
14	7		14

Egymásba ágyazott For ... Next ciklusok

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

5.8.7.2 Do ... Loop ciklus

A Do ... Loop ciklus a ciklusmagot egy feltétel teljesülésétől függően ismételteti. A variációk száma az alábbiak lehetőségek kombinációjából adódik.

A ciklusmag ismétlése történhet amíg:

- a feltétel teljesül (While, amíg),
- a feltétel igazzá nem válik (Until, mígnem).

A feltétel tesztelése lehetséges:

- a ciklusmag végrehajtását megelőzően (elől tesztelő),
- a ciklusmag végrehajtását követően (hátral tesztelő).

A Do ... Loop parancs szintaxisai:

Elöl tesztelő ciklus

```
Do {While | Until} <feltétel>  
    <utasítások>  
    <utasítások>  
    <utasítások>  
Loop
```

Hátral tesztelő ciklus

```
Do  
    <utasítások>  
    <utasítások>  
    <utasítások>  
Loop{While | Until} <feltétel>
```

Mintapéllda:

A programok Inputbox paranccsal addíg olvassák be számokat, amíg a felhasználó által megadott szám nem nulla, majd Excel munkalapon megjelenítik a beadott számokat és négyzeteiket. Az egyik program az Until, a másik a While segítségével fogalmazza meg a feltételt.

Megjegyzés:

A Do ... Loop ciklusban nincs beépített ciklus lépésszámláló, ezért egy változót kell alkalmazni, (pl. intS) a számlálásra. A bevezetett változó értékét a Do kulcsszó előtt nulla értékre kell állítani, majd a Do után növelni kell eggyel.

Program

```
Sub do_ciklus_1()  
Dim intX As Integer, intS%
```

```
intS = 0  
Do  
    intS = intS + 1  
    intX = InputBox("x=?", "Adatbekérés", 11)  
  
    Cells(intS, 1) = intX  
    Cells(intS, 2) = intX ^ 2  
Loop Until intX = 0
```

```
intS = 0  
intX = 12
```

```
Do Until intX = 0  
    intS = intS + 1  
    intX = InputBox("x= ?", "Adatbekérés", 11)  
    Cells(intS, 4) = intX  
    Cells(intS, 5) = intX ^ 2  
Loop
```

```
End Sub
```

```
Sub do_ciklus_2()
```

```
Dim intX As Integer, intS%
```

```
intS = 0  
Do  
    intS = intS + 1  
    intX = InputBox("x= ?", "Adatkérés", 11)  
    Cells(intS, 1) = intX  
    Cells(intS, 2) = intX ^ 2  
Loop While intX <> 0
```

```
intS = 0  
intX = 12
```

```
Do While intX <> 0  
    intS = intS + 1  
    intX = InputBox("x= ?", "Adatkérés", 11)  
    Cells(intS, 4) = intX  
    Cells(intS, 5) = intX ^ 2  
Loop
```

```
End Sub
```

Megjegyzés

a ciklus lépésszámlálója
hátról tesztelő Until ciklus

	A	B	C	D	E
1	11	121		7	49
2	12	144		8	64
3	13	169		9	81
4	0	0		10	100
5				0	0

a ciklus lépésszámlálója
x változó kezdő értéke

előlről tesztelő Until ciklus

	A	B	C	D	E
1	11	121		7	49
2	12	144		8	64
3	13	169		9	81
4	0	0		10	100
5				0	0

hátról tesztelő While ciklus

az x változó kezdő értéke

előlről tesztelő While ciklus

5.8.8 Programstrukturáló utasítások

A program strukturálásának alapvető célja, hogy a program fizikailag és/vagy logikailag szétbontható legyen apróbb, jól áttekinthető szegmensekre. A fizikai szétbontás komplett eljárás-könyvtárként való kezelés lehetőségét adja. A logikai szétbontás elsősorban a jobb olvashatóságot és tesztelhetőséget segíti. A VBA kétféle strukturálást ismer: eljárást és függvényt. Mindkettő képes a hívás helyén számára megadott információkat átvenni, azokon módosításokat is végezni.

5.8.8.1 Eljárás (Subrutin)

Olyan önálló program, mely hívása céljából saját azonosító névvel és az információcsere céljából kapcsolódási felülettel rendelkezik. Az eljárás specifikálására az eljárás deklarációs utasítása szolgál, ennek szintaxisa:

```
Sub <eljárásnév> (<változónév> As <adattípus>)  
    <eljárástörzs>  
End Sub
```

Az <eljárásnév> megadásakor az általános elnevezési szabályok érvényesek. Az <eljárásnév> utáni kerek zárójel közötti rész a „formális paraméterek” listája. Paraméterek, mert általuk az eljárás a hívásakor a működést szabályozó információkat kapnak. „Formálisak”, mert tényleges változókká csak az eljárás meghívásakor válnak.

A Sub eljárást hívó Call „eljáráshívó” szintaxisa:

```
Call <eljárásnév> („aktuális” paraméterek listája)
```

Az „aktuális, illetve a formális paraméter” listák elemeit vesszővel elválasztva kell írni. A hívásban szereplő aktuális és deklarációban levő formális paraméterek megfeleltetése a listabeli sorrendjük alapján történik. Az <aktuális paraméterek listája> az eljárást hívó programban a híváshelyén látható változókat, illetve kiértékelhető kifejezéseket is tartalmazhat, de a formális paraméterek listájában, csak változó szerepelhet. Az „aktuális” és „formális” paraméterek lista megfelelő paramétereinek kompatibilisnek (egymásba átkonvertálhatónak) kell lenni. Ez azt jelenti, hogy például egy integer típusú aktuális paraméter átadhatja az értékét egy integer, vagy long típusú formális paraméternek, de hibát okoz, ha a formális paraméter byte típusú és az átadandó érték nagyobb, mint 255.

E paraméter átadási mód, a címszerinti paraméter-átadás. Ez azt jelenti, hogy átadáskor nem értékelődik ki a hívásbeli aktuális paraméter, hanem annak csak memória-területe (címe) rendelődik a formális paraméterhez. Ezáltal egy formális paraméterre történő hivatkozás (az <eljárás-törzs>-ben), valójában a neki megfelelő aktuális paraméterhez rendelt memóriaterületre történik.

A Sub eljárást hívó Call parancs működését bemutató példa:

Program

```
Sub x_az_x_ediken()  
Dim intX As Integer, intY As Long
```

```
Cells(1, 1) = "x": Cells(1, 2) = "y"  
intX = InputBox("x = ?")
```

```
Call Sokadik(intX, intY)
```

```
Cells(2, 1) = intX: Cells(2, 2) = intY  
Cells(5, 1) = "x": Cells(5, 2) = "y"  
End Sub
```

```
Sub Sokadik(intXbe%, intYki As Long)  
Dim intl As Integer
```

```
intYki = 1  
For intl = 1 To intXbe  
    intYki = intYki * intXbe  
Next intl
```

```
End Sub
```

Megjegyzés

	A	B	C
1	x	y	
2	5	3125	
3			
4			

Call hívás parancssor

A For ciklus helyett az
 $\text{intYki} = \text{intXbe} \wedge \text{intXbe}$
parancs is használható.

5.8.8.2 Függvény (Function)

Az eljáráshoz képest azzal a többlettel rendelkezik, hogy neve értéket kap, ezzel képes az általa előállított információt a hívás helyére visszaadni. A függvények specifikálására a függvénydeklarációs utasítás szolgál, ennek szintaxisa:

```
Function <függvéynév> (<változónév> As <adattípus>) As <adattípus>  
    <függvénytörzs>  
End Function
```

A Sub eljárásban levő Function „eljáráshívó” szintaxisa:

<változó> = <függvéynév> („aktuális” paraméterek listája)

A <függvéynév> írására az általános elnevezési szabályok érvényesek. A <függvéynév> értéket felvevő változó is, melyre a változókra előírtak is vonatkoznak, így adattípusa is van. A < függvéynév> utáni kerek zárójelek közötti rész a „formális paraméterek” listája.

Program

```
Sub function_pelda()
```

```
Dim intA(3) As Integer, intB%(3)  
Dim intN As Integer, Sk As Single  
Dim vh1 As Single, vh2!, intI%
```

```
For intI = 1 To 3  
    intA(i) = Cells(1, i + 1)  
    intB(i) = Cells(2, i + 1)  
Next intI
```

```
intN = 3  
Sk = f(intA(), intB(), intN)  
Cells(4, 2) = Sk
```

```
vh1 = Sqr(f(intA(), intA(), intN))  
Cells(5, 2) = vh1
```

```
vh2 = Sqr(f(intB(), intB(), intN))  
Cells(6, 2) = vh2
```

```
End Sub
```

```
Function f(x%(), y%(), z%) As Single  
Dim i As Integer
```

```
For i = 1 To z  
    f = f + (x(i) * y(i))  
Next i  
End Function
```

Megjegyzés

	A	B	C	D	E
1	a	1	-2	2	
2	b	3	0	4	
3					
4		11			
5		3			
6		5			
7					

Skalárszorzat számolása

„a” vektorhossz számolása

„b” vektorhossz számolása

A program a futásakor az „=” jeltől jobbra levő kifejezést hajtja végre. A fordítóprogram konstatálja, hogy ott egy „f” Function változó van, keres tehát a Subrutinon kívül egy Function rutint, melynek neve „f”. A program futásakor a zárójelben levő „aktuális paraméterek” (a(), b(), n) értékei átadódnak az „f” Function formális paramétereinek (x(), y(), z).

A <functiontörzs> utasításainak futása alatt az „f” függvényváltozó értéket kap. Az End Function sor után a program az „f” függvényváltozó értékével visszatér a hívó Subrutin hívási sorához, ezt az értéket kapja az „=” jeltől balra levő „Sk” változó. Ezt követi „Sk” változó értékének a kiírása „B4” cellába.

6 Feladatok

6.1 Programozási mintagyakorlatok alapvető algoritmusokkal

6.1.1 „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	I
1	1	2	3	4	5	6	7	28	
2	n tömb elemeinek Összege								
3									

tömb beolvasása cellákból
egydimenziós tömb elemeinek összegzése

6.1.2 n x 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()
```

Megjegyzés

Tömb elemeinek összege

	A	B	C	D	E	F	G	H	I
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								
8									

a tömb beolvasása cellákból
a tömb elemek összegének számolása

tömboszlopok elemeinek összege

```

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
    For intJ = 1 To intN
        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
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
        sngTomb(intI, intJ) = Cells(intI, intJ)
        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%

```

```
intN = 7: intSum_atlo = 0
```

```

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

```

```

Cells(intN + 1, intN+ 1) = intSum_atlo
Cells(intN + 2,1) = "Tömb bal átlója elemeinek összege"
End Sub

```

oszlopösszegek nullázása

a tömb beolvasása cellákból

	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	5	10	15	20	25	30	35	
7	a oszlopok elemeinek Összege							

a tömb sorok elemeinek összege

	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 tömb beolvasása cellákból

a tömb sorösszegeinek számolása

tömb bal felsőből induló 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							
10								

a tömb átló összegének számolása

6.1.3 Tömbváltozó elemei közötti Min és Max értékek keresése

Tömbváltozóban megkereshetők a 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 változónak „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

```
Sub min_max()
```

```
Dim intA() As Integer
Dim intMax%, intMaxh%
Dim intMin%, intMinh%
Dim intT%, intI%
```

```
Rows(2).ClearContents
intT = 0
```

```
Do
```

```
    intT = intT + 1
```

```
    ReDim Preserve intA(intT)
```

```
    intA(intT) = Cells(1, intT)
```

```
Loop While IsEmpty(Cells(1,intT+1)) = False
```

```
Cells(2, intT) = intT
```

```
Cells(3, intT) = „Szám”
```

```
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
```

Megjegyzés

dinamikustömb deklarációja

célcellák (2. sor) előtörlése

intA() tömb méretének növelése

intA() tömb feltöltése

intMax és intMin kezdőérték megadása
az integer adattípus értékkészlete alapján

intMax és intMin keresése

	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
4											
5											

Találat kiírása, találati helyegyezéssel

6.1.4 Sorfüggvények és Faktoriális számolás rutin eljárások

6.1.4.1 Sorfüggvény példa

Adott az $f(k)=3/(k^2+5k+6)$ függvény. Megírandó a „Sorfuggveny” VBA program, amely a $\text{Sum} = \sum_{k=1}^m f(k) = f(1) + f(2) + \dots + f(m)$ 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	D
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	
9				
10				

Három tizedesre kerekít és cellába ír.

Az összeg kiírásának feltételét vizsgálja.

6.1.4.2 Faktoriális számítása

A Faktoriális programot beolvassa x és n értékeket, majd kiszámolja a $1, x, \frac{x^2}{2!}, \frac{x^3}{3!}, \dots, \frac{x^n}{n!}$ faktoriális sorfüggvény tagjainak értékét és a tagok összegét. Megfigyelhető, hogy a tagok egymásból $\frac{x}{i}$ értékkel való szorzással állíthatóak elő, ahol az i 1-től n-ig levő egész számok halmaza.

Program

Sub faktoriális()

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

	A	B	C	D
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	
15				

tagok összegzése

6.1.5 Műveletek vektorokkal és tömbökkel

6.1.5.1 Vektorok összege, szorzata, hossza

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

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

Hossz: $A \text{ hossz} = \text{Sqr}((A(1) * B(1)) + (A(2) * B(2)) + \dots (A(N) * B(N)))$

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

For intI = 1 To 3
 intE = intE + intA(intI) * intB(intI)
Next intI

Cells(9, 2) = intE
intE = 0

For intI = 1 To 3
 intE = intE + intA(intI) ^ 2
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

Megjegyzés

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

A * B számolás

A hossz négyzetének számolása

B hossz négyzetének számolása

	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		
14				

6.1.5.2 Tömbök szorzása

A tömbök szorzásának feltétele, hogy egyik dimenziójuk nagysága azonos kell legyen. A mintapélda egy $A(2 \times 3)$ tömb szorzása $B(3 \times 4)$ tömbbel, melynek eredménye $C(2 \times 4)$ 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
```

```
intA(intI, intK) = Cells(intI + 1, intK) „A” tömb feltöltése
```

```
Next intK
```

```
Next intI
```

```
For intI = 1 To 4
```

```
For intK = 1 To 3
```

```
intB(intI, intK) = Cells(intK + 4, intI + 4) „B” tömb feltöltése
```

```
Next intK
```

```
Next intI
```

```
Range("E2:F3").ClearContents
```

„C” tömb helyének törlése

```
For intI = 1 To 2
```

3x For ... Next számoló

```
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			
10								

6.1.6 Kisebb – nagyobb rendezés (Buborékredezés)

Program

```
Sub novekvo_rendezes()  
  
Dim intK%, intI%  
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
```

Megjegyzés

A program futtatása előtt véletlenszerűen ki kell tölteni az A1:A10 cellákat 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
12	

6.2 Kidolgozott gyakorló feladatok

6.2.1 Program írása blokkdiagram alapján

6.2.1.1 Térfogatszámolás feladat

Írja meg a „Térfogatszámolás” 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 $R_1 > R_2$ 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 Térfogatszámolas()

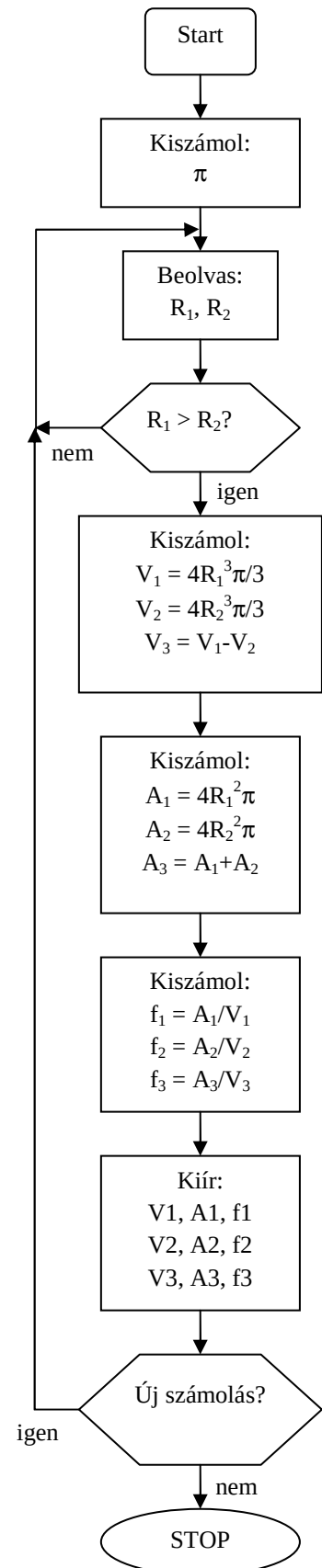
```
Dim sngR1 As Single, sngR2!  
Dim sngV1 As Single, sngV2!, sngV3!  
Dim sngA1 As Single, sngA2!, sngA3!  
Dim sngf1 As Single, sngf2!, sngf3!  
Dim strDontes As String  
Dim sngPi As Single
```

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 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
```

Ha a programot úgy kell módosítani, hogy ciklusokat, tömböket, függvényeket és a szubrutint is tartalmazzon, akkor az alábbiakat kell figyelembe venni:

- R , V , A és f értékeket lehet tömbökben tárolni, míg strDontes és sngPi változókat nem célszerű (le-

het 1 elemes tömböket deklarálni, de nincs értelme),

- az if then - goto újra 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 Terfogatszamos2()

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.

6.2.2 Gyakorlatok For - Next és Do - Loop ciklusok alkalmazására

6.2.2.1 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 nye-

resi esély a $\frac{n!}{(n-k)!k!}$ képlet segítségével számítható ki. A gyorsabb számolás érdekében a fenti képlet

alábbi egyszerűsített változatát használja: $\frac{n(n-1)\dots(n-k+1)}{k!} = \prod_j \frac{n-j}{1+j}$; ahol $j = 0, 1, \dots, k-1$. A prog-

ramok 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.

6.2.2.2 Pithagorasz-féle számhármak meghatározása

Írjon VBA programot Do-Loop (előltesztelő) és a For-Next ciklus alkalmazásával, amelynek segítségével 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 $\text{intC} \leq 100$. A két befogó (intA, intB) közül az egyik befogó (intB) hosszának nagyobbnak kell lennie, mert intA = intB esetén az $\text{intA}^2 + \text{intB}^2 = \text{intC}^2$ összefüggés értelmében $\text{intC}^2 = 2 \times \text{intB}^2$, amiből egyértelmű, hogy intC és intB egyszerre nem lehet egész szám. A fentebb leírtak alapján $\text{intA} < \text{intB} < \text{intC}$. 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 sor-

ba) 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 $\text{intA}^2 + \text{intB}^2 = \text{intC}^2$ feltétel. Amennyiben a feltétel teljesül, akkor a változók értékeit ki kell írni az Excel munkalapra, illetve intI értékét meg kell növelni, hogy a következő számhármass 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 Pithagorasz 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(intl, 1) = sngA
    Cells(intl, 2) = intB
    Cells(intl, 3) = intC
    intl = intl + 1
End If
Next intB
Next intC

End Sub
```

Az Sqr függvény a gyökvonást végzi. Természetesen jó megoldás a

$$\text{sngA} = (\text{intC}^2 - \text{intB}^2)^{0.5}$$

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 $\text{intB} > \text{Int}(\text{sngA})$ feltételre nem csak azért van szükség, mert korábban kikötöttük, hogy $\text{intB} > \text{intA}$, hanem azért is, hogy a program ugyanazt a megoldást ne kétszer találja meg ($\text{intA} = 3, \text{intB} = 4, \text{intC} = 5$; illetve $\text{intA} = 4, \text{intB} = 3, \text{intC} = 5$).

A feladat természetesen megoldható Do - Loop ciklus alkalmazásával is:

```
Sub Pithagorasz_Do()

Dim intB%, intC%, intl%
Dim sngA as Single

Cells(1, 9) = "A"
Cells(1, 10) = "B"
Cells(1, 11) = "C"

intC = 3
intl = 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(intl, 9) = sngA
            Cells(intl, 10) = intB
            Cells(intl, 11) = intC
            intl = intl + 1
        End If
        intB = intB + 1
    Loop
    intC = intC + 1
Loop
```

End Sub

Látható, hogy a Pithagorasz_Do program csak annyiban Pithagorasz_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.

6.2.2.3 Számkitaláló feladat

Írjon olyan VBA programot Do-Loop (hátralékos) ciklus alkalmazásával, amelyben a számítógép ki-gondol 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 játé-kot 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, találja ki melyik számra 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 százszal, 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 megnyi-

tása majd a tipp bekérése:

```
Do  
    intTipp = InputBox("Melyik számra 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 "Gratulalok! 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 számra gondoltam!"  
    ElseIf intTipp > intGondol Then  
        MsgBox "Kisebb számra gondoltam!"  
    Else  
        MsgBox "Gratulalok! Kitalálta a számot!"  
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 játékot kezdeni? (i = igen, bármi más = nem)", "Adatbekeres", "i")  
Loop While strDontes = "i"
```

```
End Sub
```


Természetesen a külső ciklus bezárása a

Loop Until strDontes <> "i"

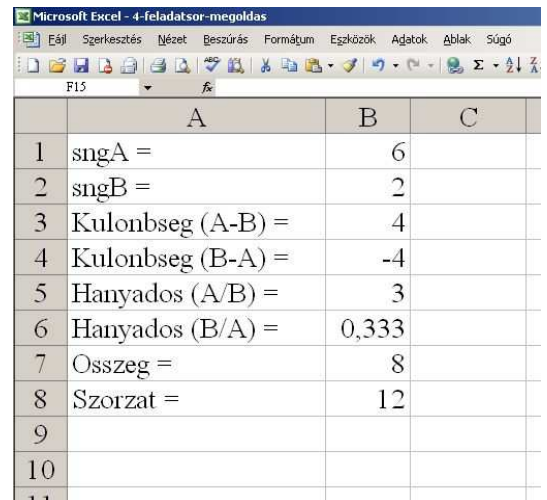
paranccsal is megtörténhet.

6.2.3 Gyakorlatok Function függvények és szubrutinok alkalmazására

6.2.3.1 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			
11			

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 értéket!", "Adatbekeres", 6)  
sngB = InputBox("Kerem B értéket!", "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 értékeit 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 $sngA \neq 0$, $sngB \neq 0$ 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 értelmezhető"
Else
    Cells(5, 2) = sngHanyados1
End If
```

```
Cells(6, 1) = "Hanyados (B/A) ="
```

```
If sngA = 0 Then
    Cells(6, 2) = "Nem értelmezhető"
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! Osztó 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 $sngAf \neq 0$ és $sngBf \neq 0$ feltételek vizsgálatát is el kell végezni. A függvény Exit Function parancsához hasonlóan ebben az esetben az Exit Sub utasítás szolgál a főprogramba történő visszatérésre.

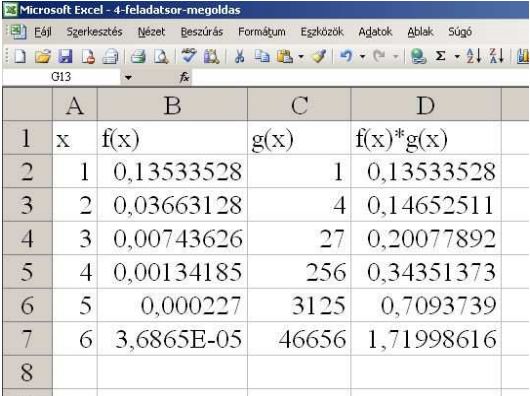
6.2.4 Fájlműveletek gyakorlatai

6.2.4.1 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

$$f(x) = x * e^{-2x} \quad \text{és} \quad g(x) = x^x$$

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” fileba. (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 intl 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)"

intl = 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)
    intl = intl + 1
    Input #1, intBe
    Cells(intl, 1) = intBe
    intVeg = intl
Loop
```

Close #1

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

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

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)
    intl = intl + 1
    intVeg = intl
    Input #1, intBe
    Cells(intl, 1) = intBe
    Cells(intl, 2) = Exponencialis(intBe)
    Cells(intl, 3) = Hatvany(intBe)
    Cells(intl, 4) = Cells(intl, 2) * Cells(intl, 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 nagyobbak kell lennie, mint az intBef változó integer értéktartománya. Ennek a követelménynek megfelel a long adattípus.

6.3 Önállóan megoldandó gyakorló feladatok

6.4 Korábbi zárthelyi feladatsorok

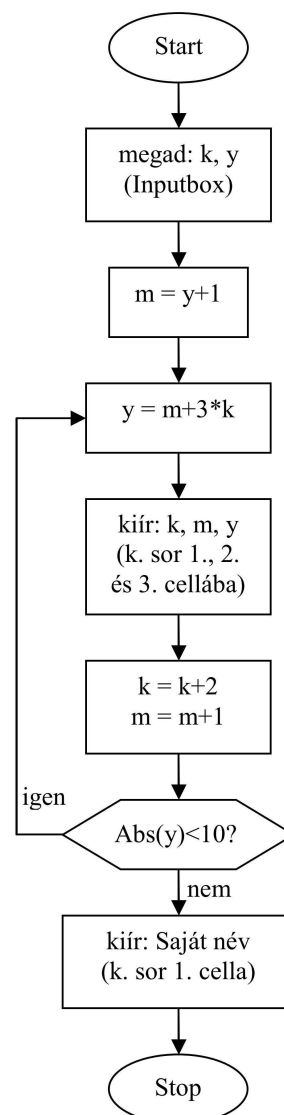
6.4.1 2009 tavaszi feladatsorok

6.4.1.1 1. feladatsor

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/dm³)” 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 „Solver”-t használva úgy számítson ki új „a” és „b” értékeket, hogy J8 cella értéke a lehető legkisebb legyen. A megoldás megtalálásakor a Solverrel szűrassa 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								

1. digi – 2009. tavasz



6.4.1.2 2. feladatsor

2. díger – 2009. tavasz

1. Adott az $f(k)=3/(k^2+5k+6)$ függvény. Írja meg a „Megoldas1” VBA

programot a $\text{Sum} = \sum_{k=1}^m f(k) = f(1) + f(2) + \dots + f(m)$ 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 kiírja.

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 kiírása a „Munka1” munkalapon történjen és az itt megadott képnek (mely $m=5$ és $p=3$ esetét mutatja) megfelelő szerkezetű eredményt adja!

	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	

(6 pont)

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 $\text{Sum} \leq 0.6$ feltétel teljesül? Indokolja meg választát! (1 pont)

6.4.1.3 3. feladatsor

3. díger – 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:

$$\text{lngTomb}(i) = \text{intTomb}(i) \times k,$$

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

$$\text{lngOsszeg} = \sum_{i=1}^n \text{lngTomb}(i)$$

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

6.4.1.4 4. feladatsor

4. díger (pótdíger) – 2009. tavasz

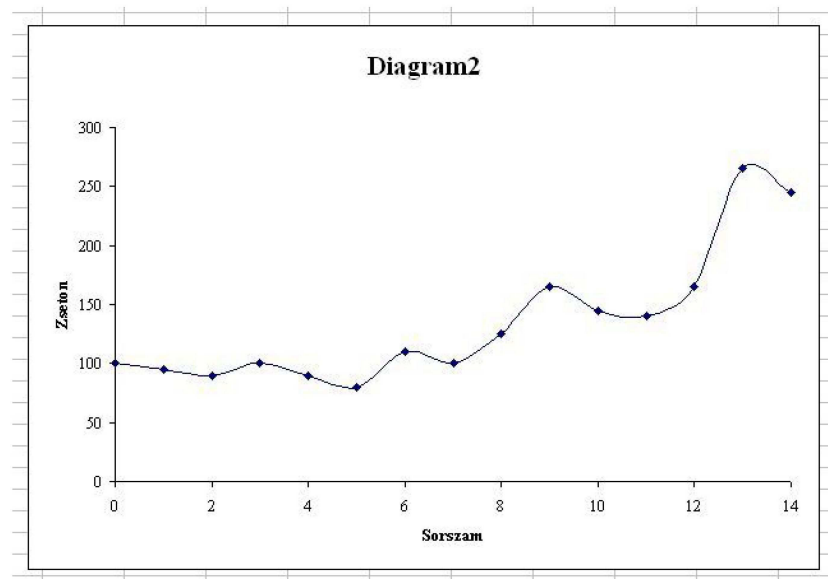
- Írjon a „Module 1” modulba VBA programot, amely megnyitja a „Poker-be.txt” file-t, először beolvassa kiindulási zsetonösszeget az integer típusú „intKiindulasi” paraméterbe, majd az „intSorszam()”, „intBefizetes()” és az „intNyeremeny()” dinamikus tömbökbe (integer típusúak) beolvassa a játék sorszámát (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:

$$\text{lngOsszPenz}(\text{intI}) = \text{lngOsszPenz}(\text{intI}-1) - \text{intBefizetes}(\text{intI}) + \text{intNyeremeny}(\text{intI})$$

(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



6.4.2 2009 őszi feladatsorok

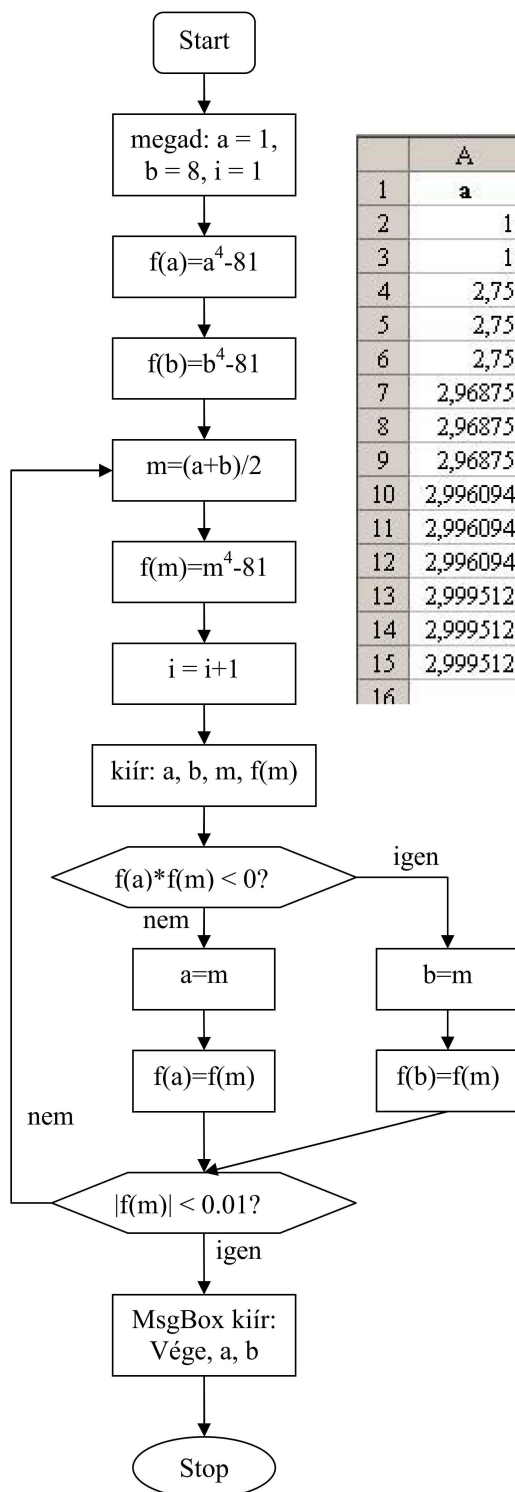
6.4.2.1 1. feladatsor, A csoport

Név:

Tankör:

Diger-1, A csoport – 2009 őszi

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					

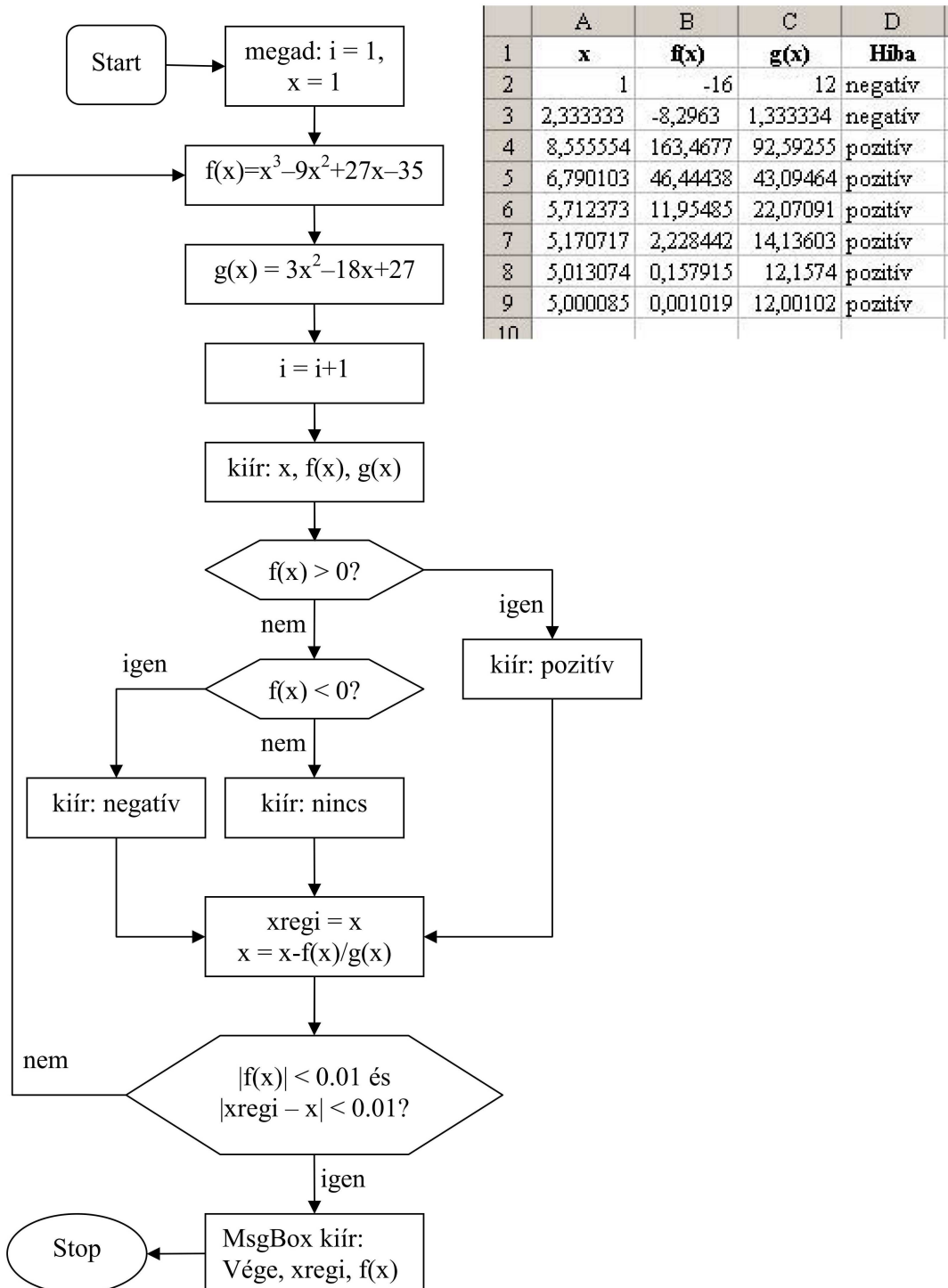
6.4.2.2 1. feladatsor, B csoport

Név:

Tankör:

Diger-1, B csoport – 2009 őszi

1. Az alább megadott blokkdiagram alapján készítse el a „Feladat_1b” programot a Do - Loop While, vagy Do - Loop Until ciklus utasítás használatával az „ $x^3 - 9x^2 + 27x - 35 = 0$ ” 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).



6.4.2.3 2. feladatsor, A csoport

Név:

Tankör:

Diger-2, A csoport – 2009 őszi

- 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:

- vagy az „Oldalhossz.txt” file-ba az alábbi módon: ABhossz = érték, AChossz = érték, BChossz = érték
- 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	legnagyobb	
3	BChossz = 5,6569		

6.4.2.4 2. feladatsor, B csoport

Név:

Tankör:

Diger-2, B csoport – 2009 őszi

- Írja meg a „Munkaber” nevű VBA programot, amely először For-Next ciklus alkalmazásával olvassa be az „Dolgozok.txt” file-ból a „Név” 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:

$$\text{Kifizetés} = \text{órabér} \times \text{ledolgozott órák száma} + \text{jutalom},$$

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:

- vagy az „Osszegzes.txt” file-ba az alábbi módon: Név = kifizetendő összeg,
- vagy a munkalap celláiba az alábbi ábrán látható módon. Ebben az esetben figyeljen rá, hogy a „B” oszlopban a kifizetendő 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	

6.4.2.5 3. feladatsor, A csoport

Név:

Tankör:

Diger-3, A csoport – 2009 ős

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” értékének 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	Adatszám =	8
2	Számok =	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

6.4.2.6 3. feladatsor, B csoport

Név:

Tankör:

Diger-3, B csoport – 2009 őszi

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” abszolútértéke 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

6.4.3 2010 őszi feladatsorok

6.4.3.1 1. feladatsor, A csoport

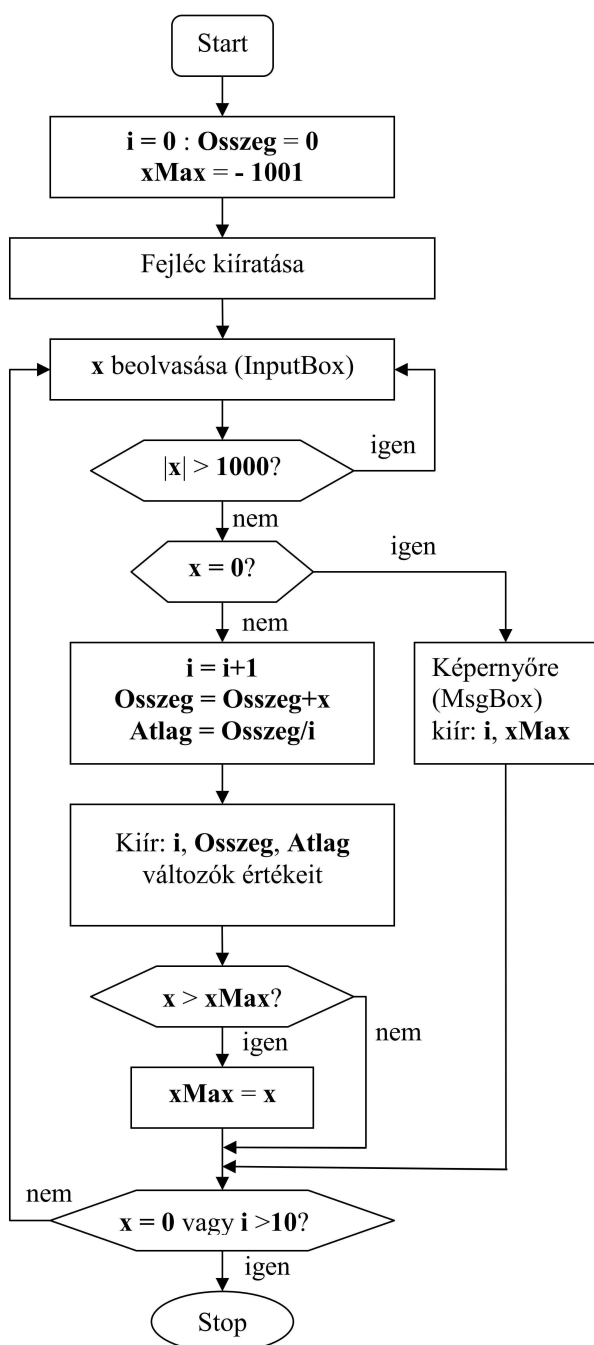
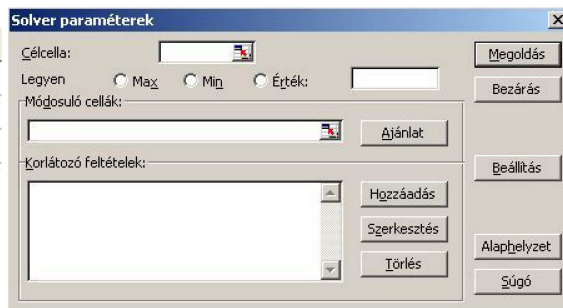
Név:

Tankör:

Diger-1, A csoport – 2010 őszi

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 írunk 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 kiíratásnak 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

6.4.3.2 1. feladatsor, B csoport

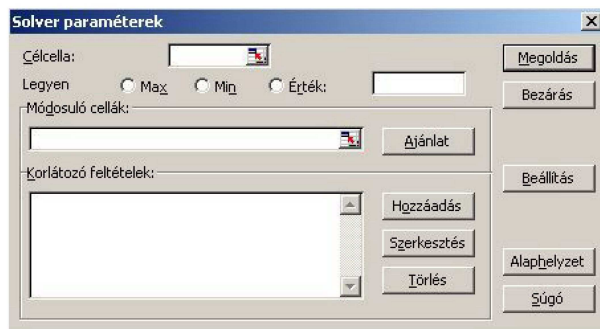
Név:

Tankör:

Diger-1, B csoport – 2010 őszi

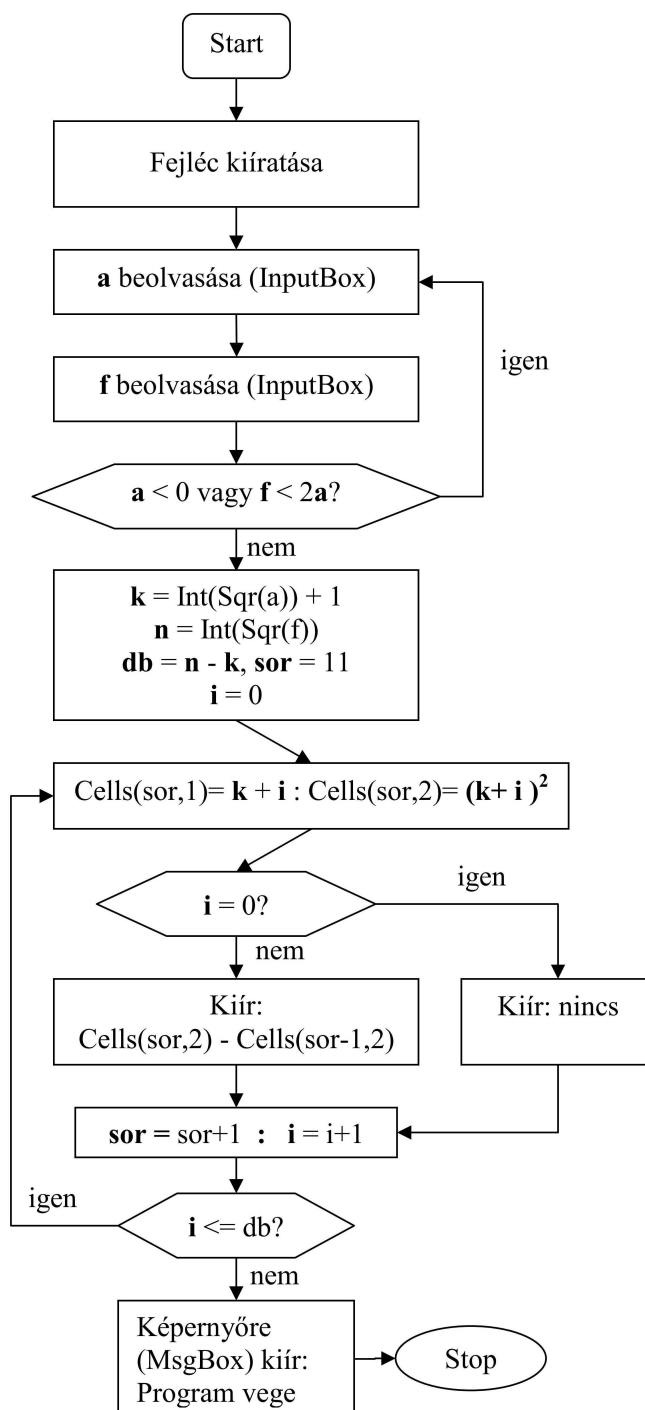
1. Az $y = a \cdot x^2 + b \cdot x + c$ másodfokú egyenlet „a”, „b” és „c” együtthatóiról tudjuk, hogy $a + b + c = 9$ és $a = b/4$. A Solver segítségével azokat az együtthatókat keressük, melyeknél az egyenlet $D = b^2 - 4ac$ diszkriminánsa zérus. Adja meg, hogy milyen képletet kell írunk a B2, B4 és 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 készítsse 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”, „(k + i)²” és „(k + i)² - (k + 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



6.4.3.3 2. feladatsor, A csoport

Név:

Tankör:

Diger-2, A csoport – 2010 őszi

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

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

6.4.3.4 2. feladatsor, B csoport

Név:

Tankör:

Diger-2, B csoport – 2010 őszi

Fájl Szerkesztés

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** x **db** méretű **Matrix** nevű Double típusú dinamikus tömböt. A **Matrix** tömb elemeinek az értékét az alábbi módon számítsa ki:

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

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	X0	Y0		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								

6.4.3.5 3. feladatsor, A csoport

Név:

Tankör:

Diger-3, A csoport – 2010 őszi

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	A	B	C	D	E
Andras,5				Jegy	Db
Balazs,4	Andra	5		1	3
Cecilia,2	Balaz	4		2	5
Dezso,1	Cecil	2		3	8
Elemer,3	Dezs	1		4	6
Ferenc,3	Elem	3		5	2
Gabor,3	Ferer	3			
Hedvig,2	Gabo	3			
Ilona,4	Hedvi	2			
Judit,1	Ilona	4			
Kalman,5	Judit	1			
Laszlo,2	Kalm	5			
Melinda,3	Laszl	2			
Nora,1	Melin	3			
Oszkar,3	Nora	1			
Peter,4	Oszk	3			
Ramona,3	Peter	4			
Sandor,3	Ram	3			
Tibor,2	Sand	3			
Ubul,3	Tibor	2			
Xaver,4	Ubul	3			
Yvonne,4	Xaver	4			
Vajk,4	Yvonn	4			
Zoltan,2	Vajk	4			
	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 fejléctet, 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)

6.4.4 2011 őszi feladatsorok

6.4.4.1 1. feladatsor, A csoport

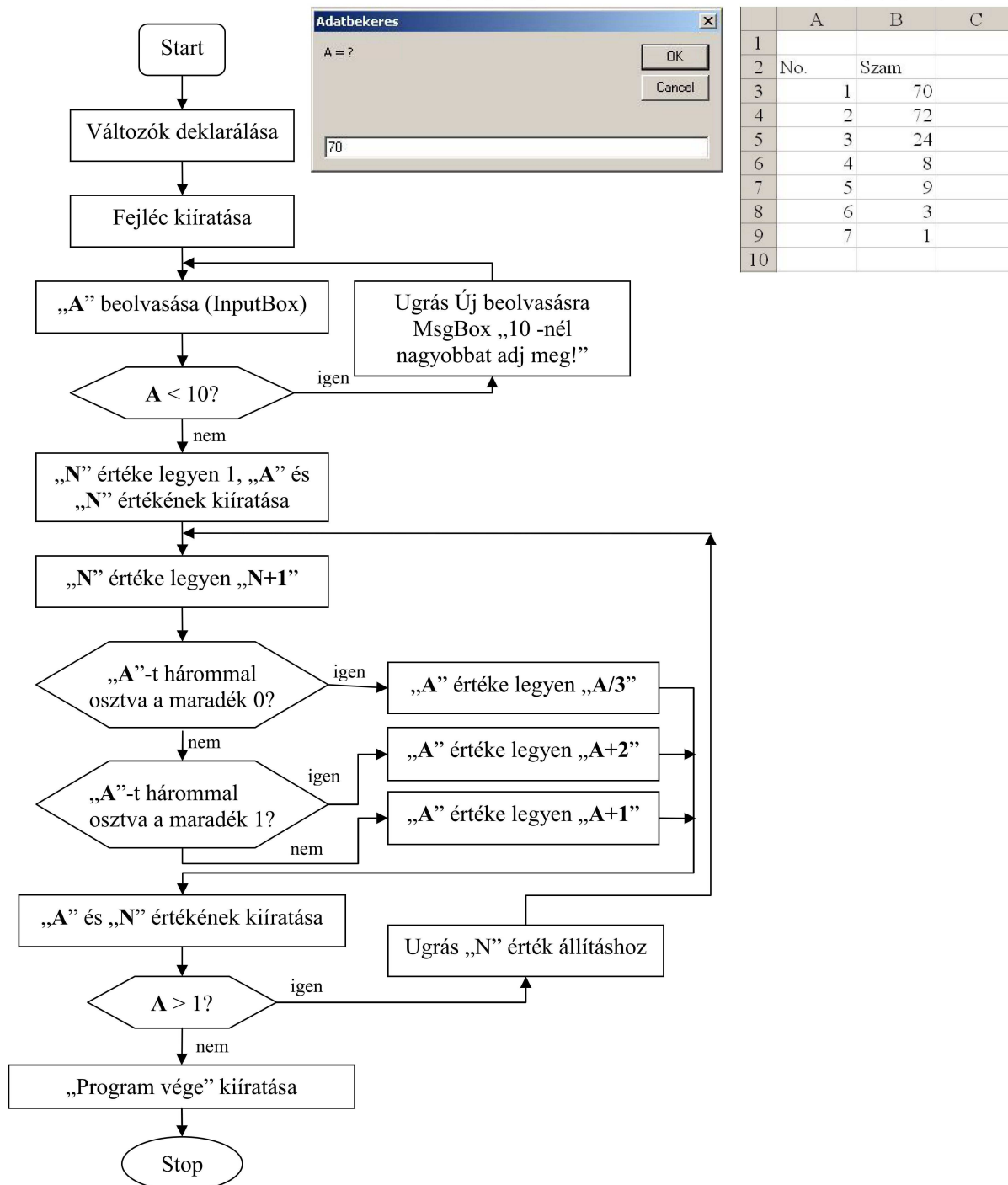
Név:

Tankör:

Diger-1, A csoport – 2011 őszi

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 vége**” szöveget a MsgBox parancssal 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)



6.4.4.2 1. feladatsor, B csoport

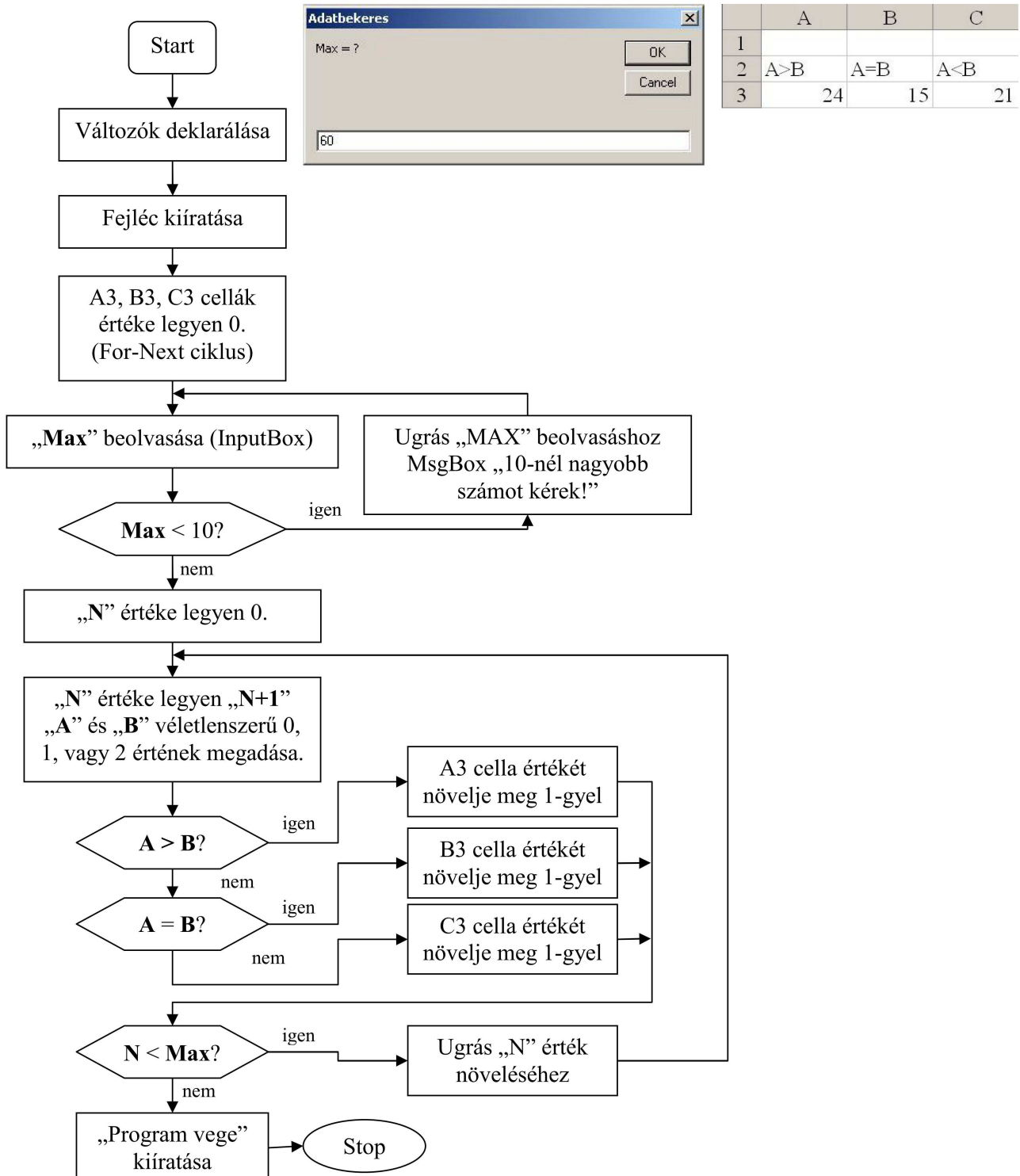
Név:

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 parancssal 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)



6.4.4.3 2. feladatsor, A csoport

Név:

Tankör:

Diger-2, A csoport – 2011 ős

- 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					

Fájl	Szerkesztés	Beállítás
1,-2.584624E-02		
2,-4.847878E-02		
3,-.1026301		
4,-.2604988		
5,-.8986193		
6,-6.023337		
7,-982.6918		
8,1980.198		
9,-982.6918		
10,-6.023337		
11,-.8986193		
12,-.2604988		
13,-.1026301		
14,-4.847878E-02		
15,-2.584624E-02		
16,-1.503035E-02		
17,-.0093325		
18,-6.099343E-03		
19,-4.154019E-03		
20,-2.926659E-03		

6.4.4.4 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 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

Fájl Szerkesztés Beállítások Súgó

"Matrixelemek osszege:"
4416

6.4.4.5 3. feladatsor, A csoport

Név:

Tankör:

Diger-3, A csoport – 2011 ős

- 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 $\text{also} \leq \text{Tomb}(i) \leq \text{felso}$, 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 hatar	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					

6.4.4.6 4. feladatsor, A csoport

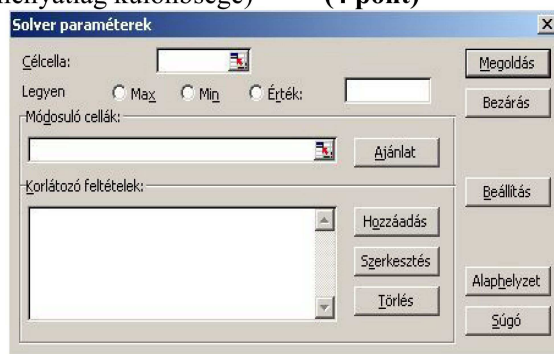
Név:

Tankör:

Diger-4, A csoport – 2011 ős

1. A 2010. 09. 28-i Zalaegerszeg-Videoton labdarúgó-mérkőzés eredményére fogadtunk úgy, hogy az elérhető fogadóirodák közül 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)**

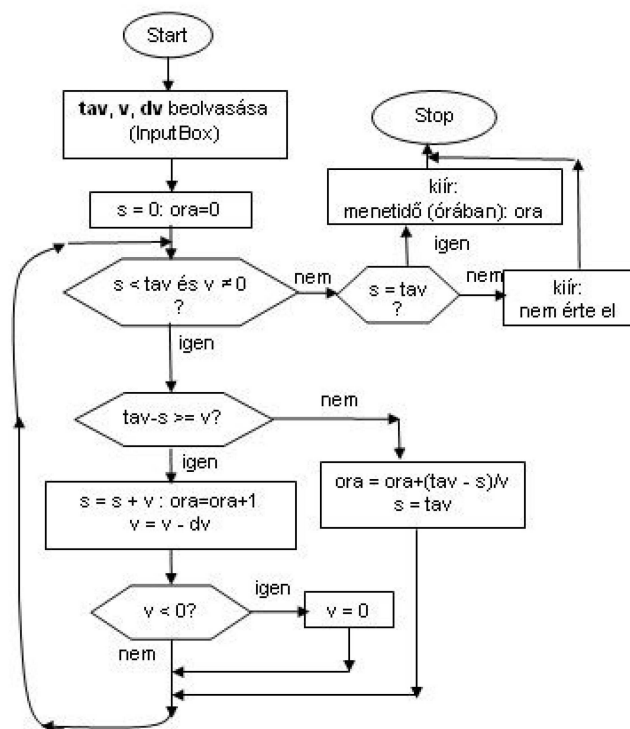
	A	B	C	D	E	F
1						
2		Tét	Szorzó	Nyeremény		Hibanegyzet
3	Zalaegerszeg		3,20			
4	Döntetlen		3,59			
5	Videoton		2,40			
6						
7			Átlag =		Összesen =	



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 $A = 2r^2\pi + 2r\pi m$ és $V = r^2\pi m$. 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 értékeit írassa 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 $0 \leq a < b \leq n$ feltételek teljesülését. (Ha nem teljesülnek, a beolvasatást 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

(12 pont)

Fájl	Szerkeszté
0,	9
1,	7.5
2,	6
3,	4.5
4,	3
5,	1.5
6,	2.25
7,	3
8,	3.75
9,	4.5
10,	5.25
11,	6

6.4.5 2012 őszi feladatsorok

6.4.5.1 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
```

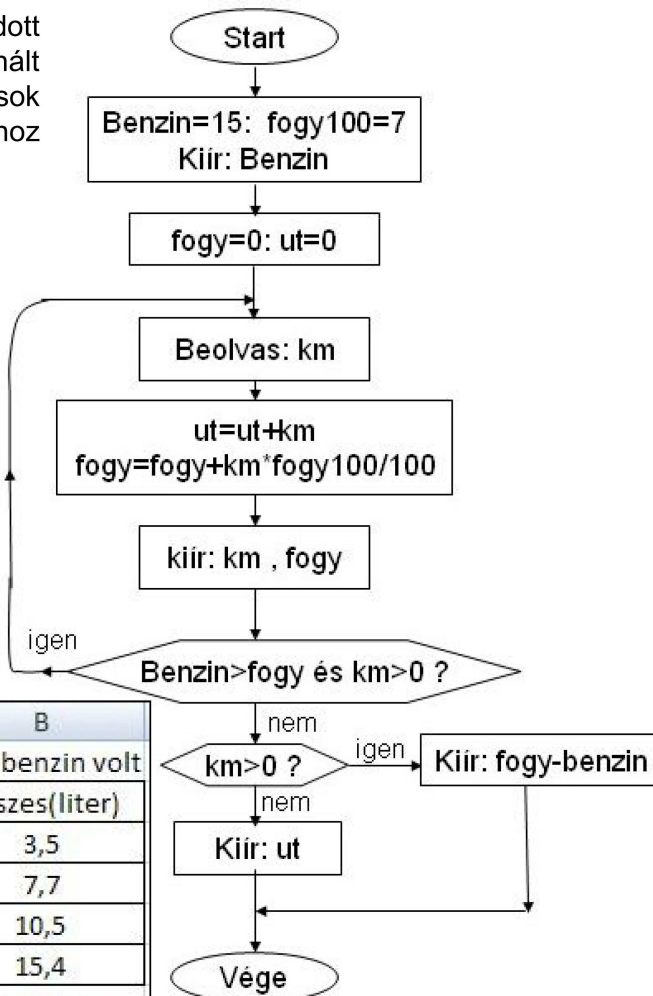
	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

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 kiírások (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 kiíratást 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 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		Megérkezettél!

	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



6.4.5.2 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.)

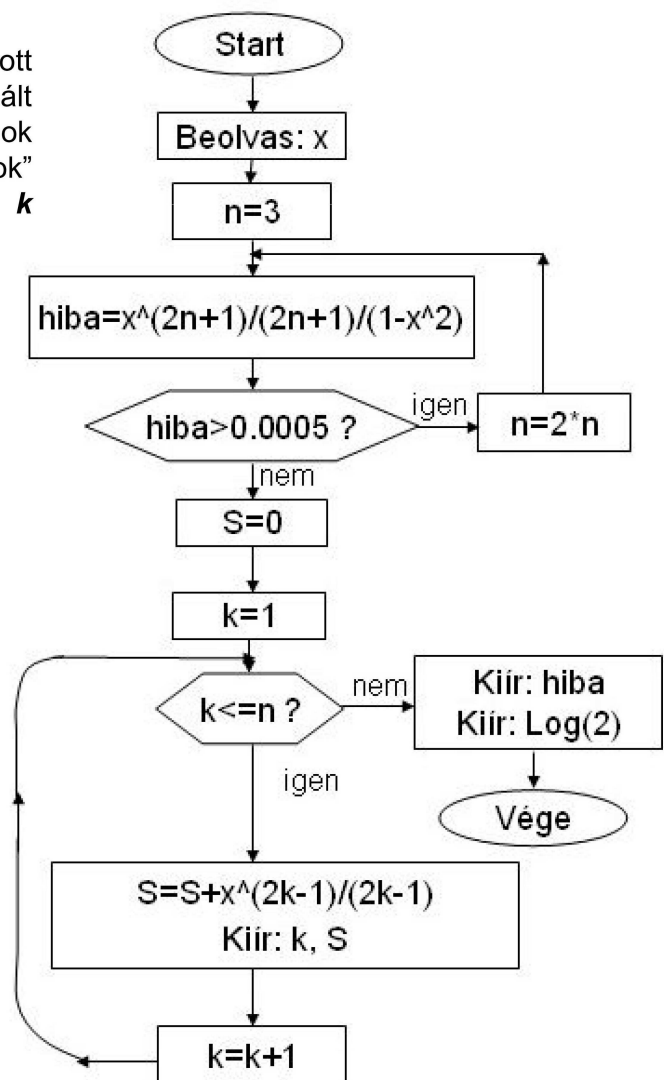
```
Sub F1_B()
Dim x As Double, y#, m%
  x = 1 + 3 * 0.2 ^ 2: m = 5
  Cells(1, 1) = "x=": Cells(1, 2) = x
  újra:
    m = m - 1
    y = 3 * (m + 1) / 2
    Cells(2, m) = m: Cells(3, m) = y
  If m - x > 1 Then GoTo újra
  m = m - 1
  Cells(2, m) = "m": Cells(3, m) = "y"
End Sub
```

	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 kiírások (az oszlopok fejlécét és az egyéb szöveges részeket is beleértve) az alábbi ábra szerint történjenek. (Az ábra az $x=0,6$ input értékhez tartozó kiírásokat 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



6.4.5.3 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** fájjal 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 kiírására 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 értékeit 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** pontot 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 kiírására, valamint az adományok összegzésére, és ezen összegek kiírására.

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 kiírás 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** oszlopaik adják.

(7,5 pont)

6.4.5.4 2. feladatsor, B csoport

Név:

tk:

Z1 (2012. nov.29.)

B

A jobbra látható **Pogacs.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 **Pogacs.txt** fájljal 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 kiírására 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)**

6.4.5.5 3. feladatsor, A csoport

Név:

Tankör:

Diger-3, A csoport – 2012 ős

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 menettérít autót 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 Beállítások

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

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

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

6.4.5.6 3. feladatsor, B csoport

Név:

Tankör:

Diger-3, B csoport – 2012 ős

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 „Kiadás” (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 „Tétel” változóba (sztring) a keresendő tétel nevét.
c) Ezután a program keresse ki a „Aru” dinamikus tömb elemei közül azokat, amelyekre igaz, hogy Aru(i) = Tétel („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

BKU bérlet, 9800
újság, 175
élelmiszer, 2870
telefonszámla, 3200
tisztítószert, 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

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

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

6.4.5.7 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) újra
- 3) Step
- 4) Next j
- 5) ()
- 6) InputBox
- 7) %
- 8) Next i
- 9) GoTo újra
- 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 „Részlet” változókbá (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 kamatthez és vonja le a törlesztőrészt. Majd a „Torlesztés” változó (double) értékét pedig növelje meg a törlesztő összeg értékével. Ezt a ciklust addig csinálja, amíg a „Részlet” nagyobb nem lesz a „Maradek” változónak kamatthez megnövelt értékénél. A program közben számolja a törlesztési hónapok számát („Honap” változó, integer). Ha a havi részlet nagyobb a „Maradek” változónak kamatthez megnövelt értékénél, a „Torlesztés” 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)

Változók és jelentésük:

Hitel: a teljes felvett pénz mennyiség (double),

Kamat: a havi kamat (double),

Részlet: a havi törlesztés összege, vagyis a havi befizetés (double),

Torlesztés: a visszafizetett pénz összege (double),

Maradek: a teljes, még vissza nem fizetett pénz összege (double),

Honap: hónapok száma, amelyekben befizetés történt (integer).

	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

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” változóiban.

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 kiírásakor 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)

Fájl	Szerkesztés	Beállítások	Súgó
Tibi, Évi, Zoli, Kati			
14,	6,	6,	0
18,	9,	19,	6
19,	24,	25,	10
20,	29,	40,	15
25,	32,	55,	18
28,	32,	58,	38
28,	34,	60,	60
28,	40,	62,	78
32,	42,	76,	84
46,	44,	86,	84
48,	50,	90,	98
52,	66,	95,	99
66,	68,	103,	101

	A	B	C	D	E	I
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				

7 Függelék

A függelék olyan Visual Basic for Application ismereteket tartalmaz, melyek nem szükségesek a tárgy teljesítéséhez, de az egyetemi évek alatt kiadott feladatok programozással történő megoldása során hasznosak lehetnek a hallgatók számára.

7.1 Rekordok és használatuk

7.2 A For Each – Next ciklus és alkalmazása

7.3 A Solver alkalmazása a programírás során

7.4 A „User Form” használata