

# VBA programozás Excelben

Wagner György  
Általános Informatikai Tanszék

# Miért programozzuk Excelben?

- Mert az elkészült program átadható másoknak, és programozói tudás nélkül is élvezhetik annak előnyeit.
- Mert időt lehet vele megspórolni (ismétlődő feladatok esetében vagy iterációs megoldásoknál).
- Mert csökken az elkövethető hibák száma (ha a program tesztelten hibátlan, onnantól kezdve minden további futtatása is hibátlan lesz).
- Mert „szabvány” teremtő lehet. Nagyobb cégeknél ugyanazt a feladatot valószínűleg többen többféle módon oldják meg. Ekkor a számítási sorrend változása miatt az eredmények eltérhetnek.
- Mert kapcsolatot lehet teremteni más alkalmazásokkal (ez persze program nélkül is megy, de azzal egyszerűbb).

# Az adatok lehetnek ...

a programon kívül (pl.: Excel táblában)

a programon belül (pl.: változókbán)

# Az utasítás

- A számítógép egyelőre nem ért emberi nyelven (igazán), ezért:
- az utasításokat valamilyen (programozási) nyelven kell leírni.
- Programozási nyelvek pl.: C, Pascal, Fortran, Assembly, PL/1, APL, Cobol, Java, Eiffel, SmallTalk, PHP, Basic, ...
- Vannak nem programozási nyelvek is: HTML, XML, SAML, ...

# VBA

- Az Excelben a VBA nyelvet lehet használni:

Visual Basic for Application

- A Microsoft Visual Basic-re épül
- Jellemzően Microsoft Office alkalmazásokhoz fejlesztették ki (Excel, Word, PowerPoint, Outlook)

# Könyvek VBA programozáshoz

## Office 2016-hoz:

- Michael Alexander: **Excel 2016 Power Programming with VBA**
- Richard Mansfield: **Mastering VBA for Microsoft Office 2016**
- John Walkenbach: **Microsoft Excel 2016 Bible**
- Jelen, Syrstad: **Excel 2016 VBA and Macros**

## • Office 2019-hez:

- Michael Alexander: **Excel 2019 Power Programming**
- Bill Jelen, Tracy Syrstad: **Microsoft Excel 2019 VBA and  
Macros**

A fejlesztőrendszer alapesetben nem érhető el.  
Be kell kapcsolni!

The screenshot shows the 'Az Excel beállításai' (Excel Options) dialog box. The left sidebar has 'Speciális' (Advanced) selected. The 'Menüszalag testreszabása' (Customize Ribbon) tab is active. The 'Fő lapok' (Main Tabs) list on the right includes 'Fejlesztőeszközök' (Developer Tools), which is checked and highlighted with a red circle. A red arrow points from the yellow text at the top to this circle. Another red arrow points from the 'Fejlesztőeszközök' label to the 'Fő lapok' list. The 'Gyakori parancsok' (Popular Commands) list on the left shows various Excel functions. The bottom of the dialog has buttons for 'OK' and 'Mégse' (Cancel).

Általános

Új

Megnyitás

Mentés

Mentés másként

Nyomatás

Megosztás

Exportálás

Közzététel

Bezárás

Fiók

Beállítások

Az Excel beállításai

Általános

Képletek

Nyelvi ellenőrzés

Mentés

Nyelv

Speciális

Menüszalag testreszabása

Gyorselérési eszköztár

Bővítmények

Adatvédelmi központ

A menüszalag testreszabása

Választható parancsok helye: i

Gyakori parancsok

Alakzatok

Az összes frissítése

Beillesztés

Beillesztés

Betűméret

Betűméret csökkentése

Betűméret növelése

Betűszín

Betűtípus

Cellaegyesítés

Cellák beszúrása...

Cellák formázása

Cellák törlése...

Csökkenő sorrend

Egyéni sorrend...

E-mail

Feltételes formázás

Formátummásoló

Függvény beszúrása...

Gyors nyomtatás

Helyesírás...

Irányított beillesztés...

Ismétlés

Kép beszúrása

Kimutatás

Kitöltőszín

Kivágás

Középre vízszintesen igazítás

Makrók

Másolás

Felvétel >>

<< Eltávolítás

Menüszalag testreszabása: i

Fő lapok

Fő lapok

☒ Kezdőlap

☐ Vágólap

☐ Betűtípus

☐ Igazítás

☐ Szám

☐ Stílusok

☐ Cellák

☐ Szerkesztés

☒ Beszúrás

☒ Lapelrendezés

☒ Képletek

☒ Adatok

☒ Véleményezés

☒ Nézet

☒ Fejlesztőeszközök

☒ Bővítmények

☒ Team

☒ Háttér eltávolítása

Új lap

Új csoport

Átnevezés...

Egyéni beállítások: Alaphelyzet i

Importálás/exportálás i

OK

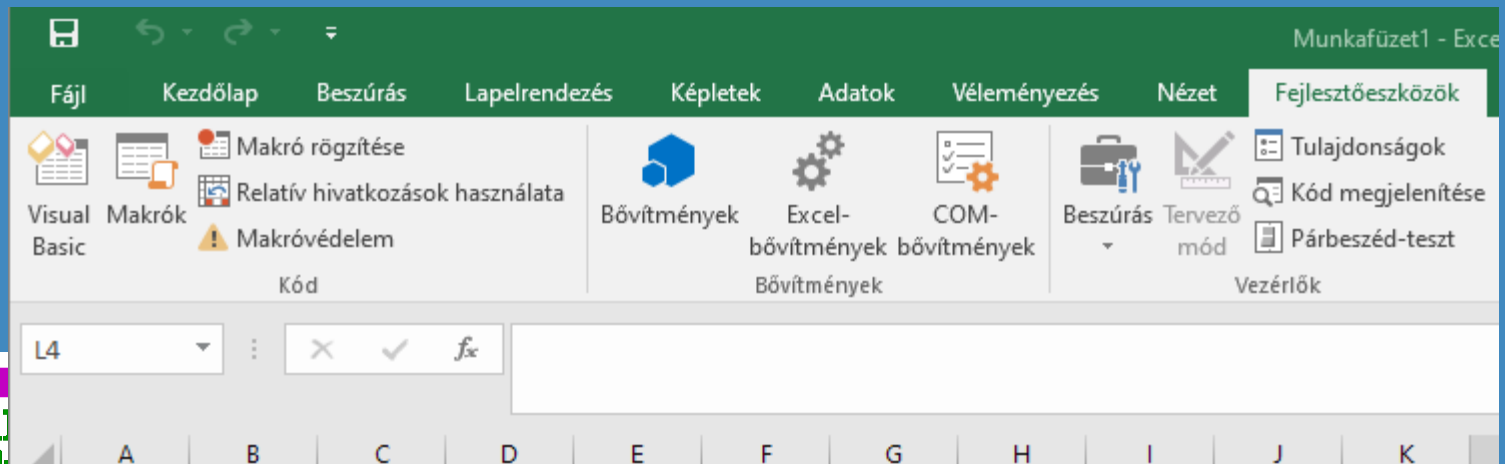
Mégse

# VBA - E

- A program írásához rendelkezésre áll egy (szöveg) szerkesztő, a VBA Editor

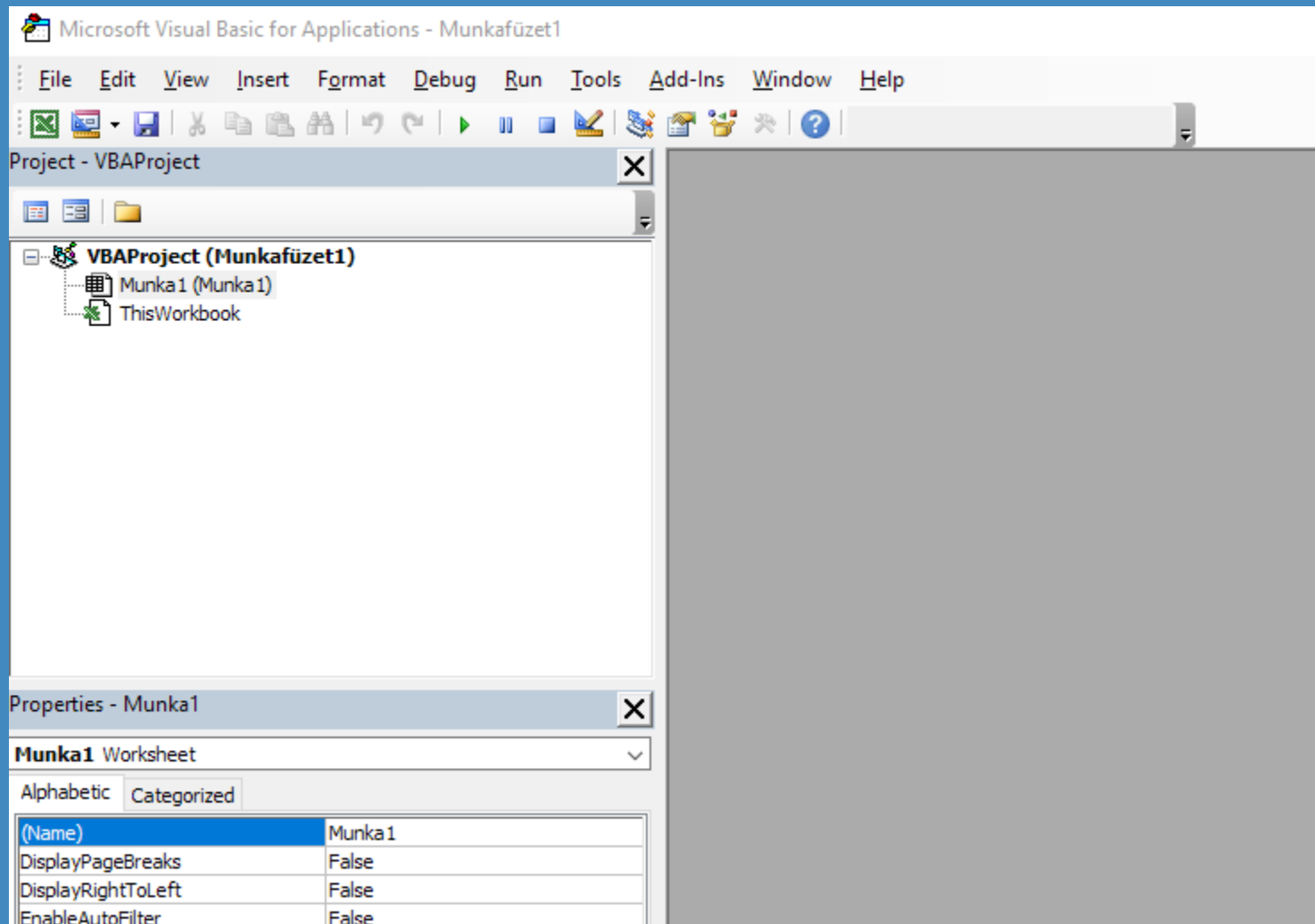
Indítása:

- ALT-F11
- Vagy Fejlesztőeszközök → Visual Basic





# VBA Editor



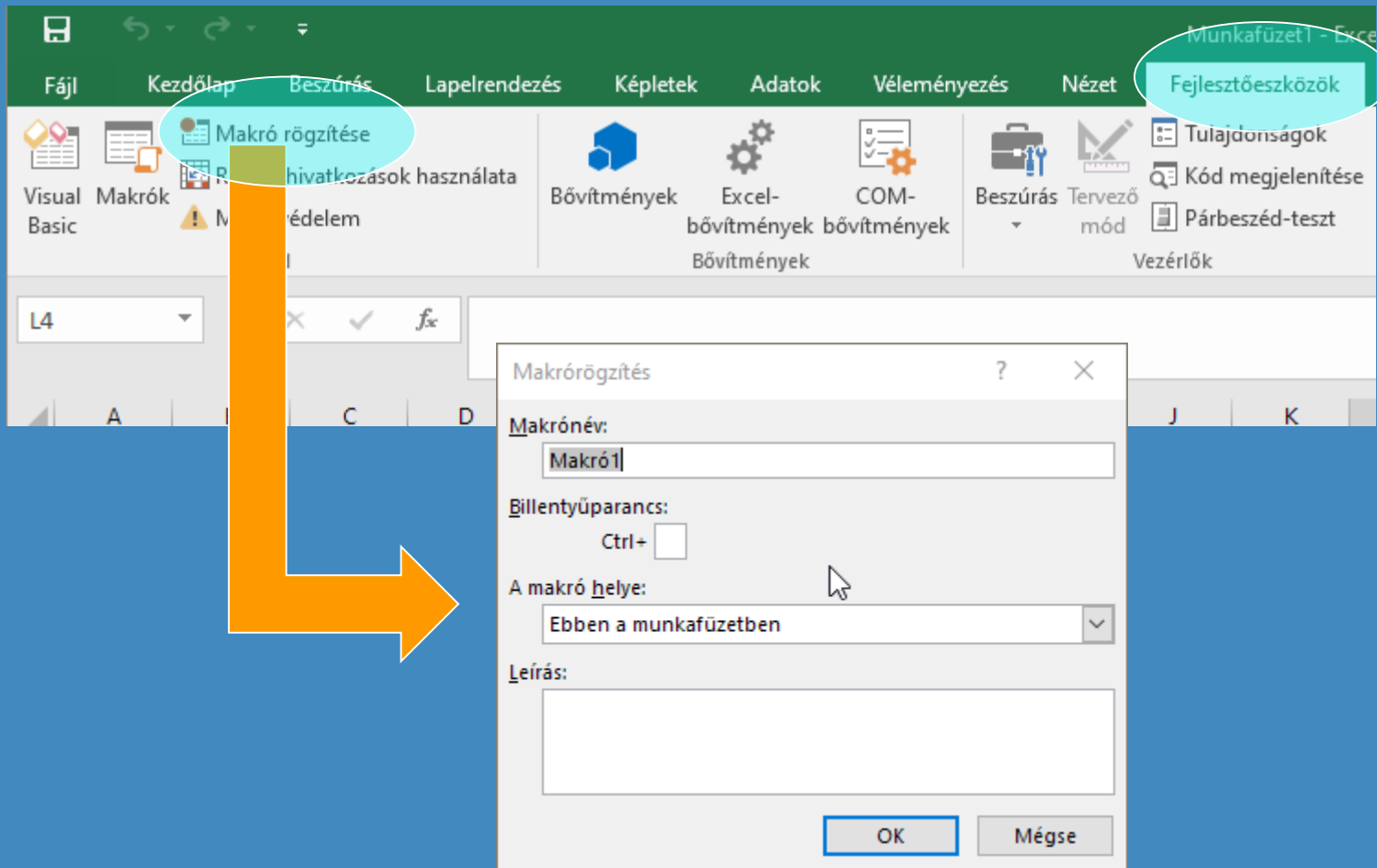
# Makró

Az Excel-ben lehetőség van a műveletek (kiválasztás, menü használat, billentyűzet, egér használat) „felvételére”, majd visszajátszására.

Valójában ekkor a műveleteket program utasításokkal helyettesíti be az Excel.

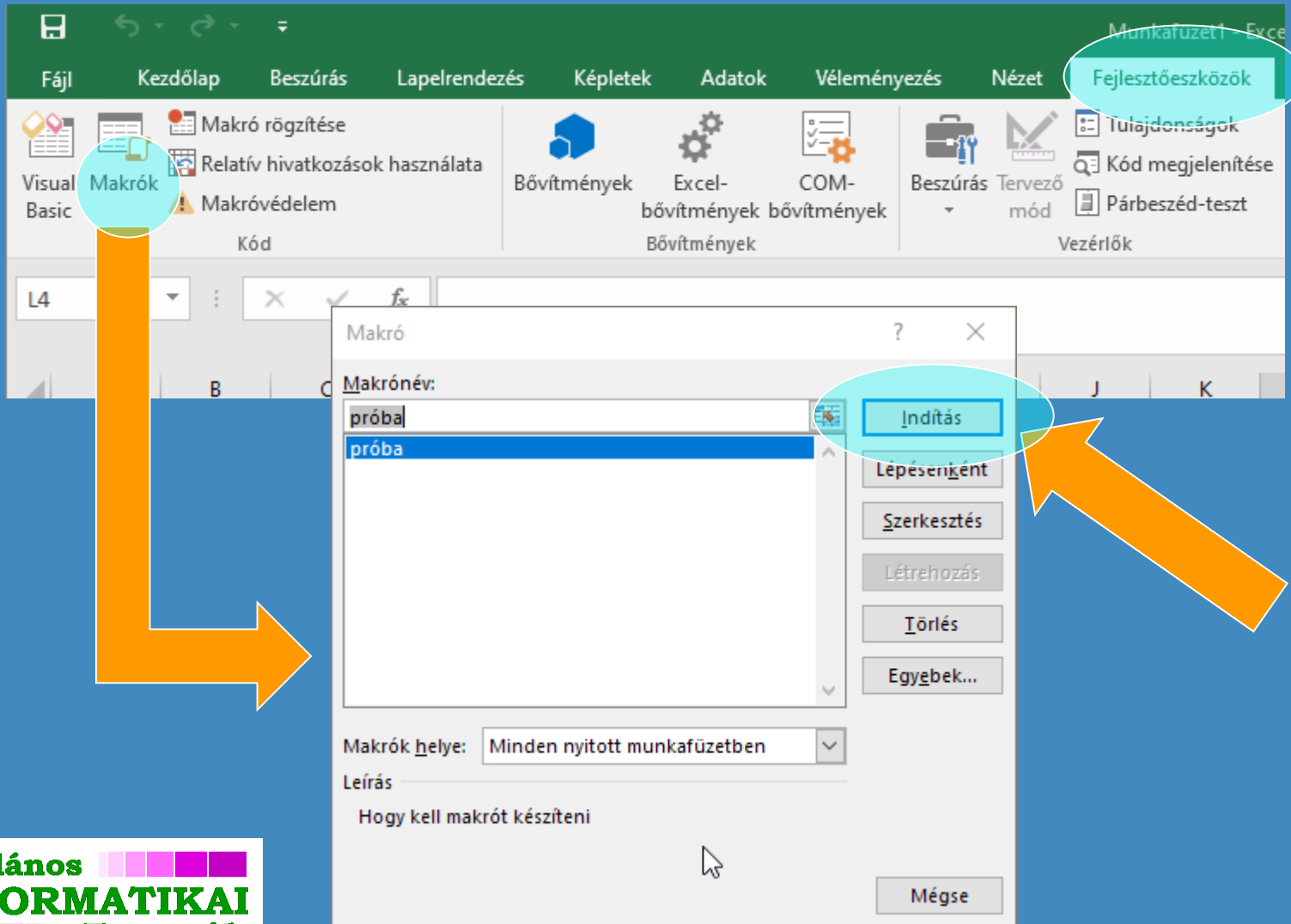
# Makró felvétele

Fejlesztőeszközök → Makró rögzítése



# Makró lejátszása

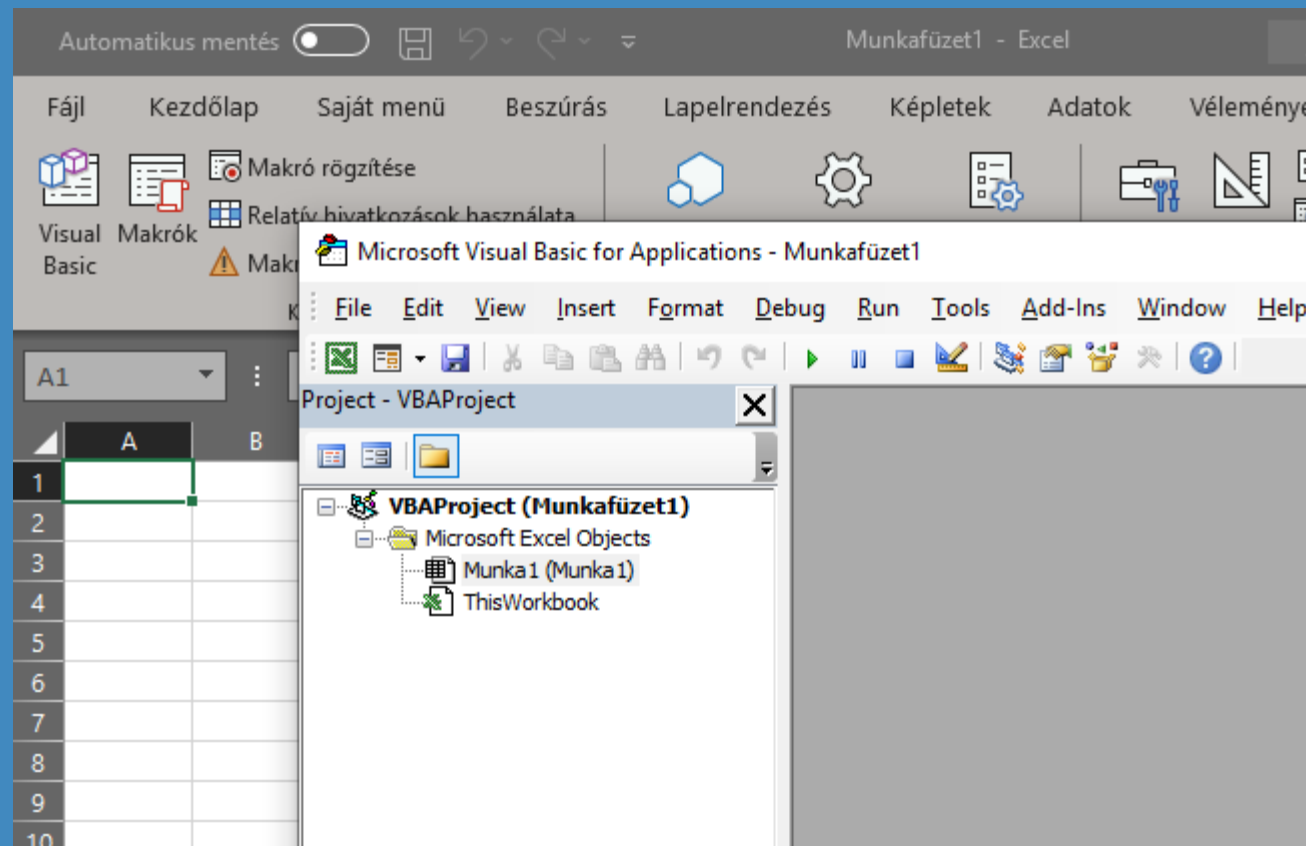
- Fejlesztőeszközök → Makrók → Indítás



# A keretrendszer (VBA-E)

Még ha az Excel magyar is, a fejlesztő-rendszer angol nyelvű:

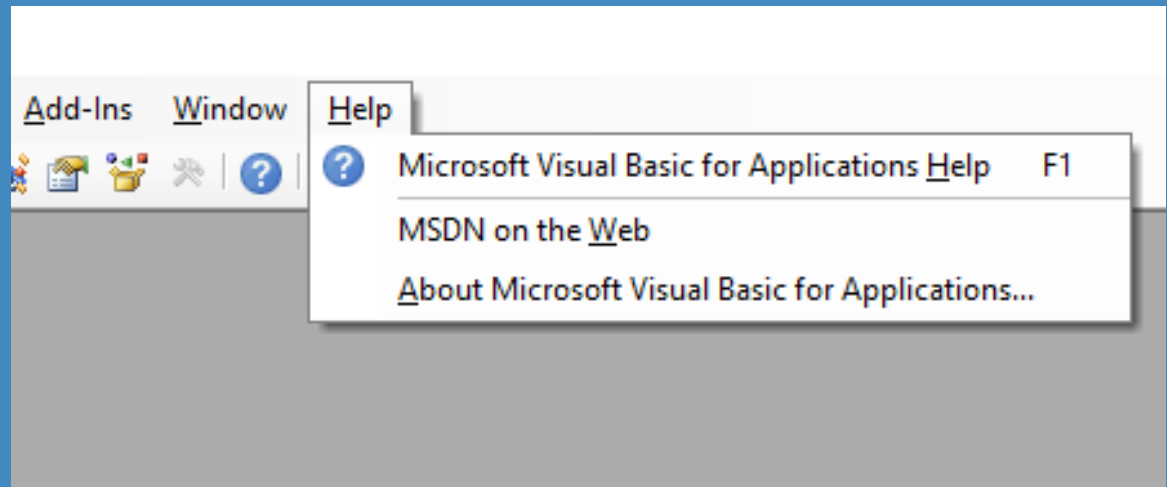
- a menürendszer,
- az üzenetek,
- Az ablakok tartalma



# A súgó

Indítása:

- F1, vagy
- Help → MSDN on the Web (Internet kell hozzá)



# Kedvcsináló

## Az első program

- Az első program szinte mindig egy kis program szokott lenni, a

”Hello World!”

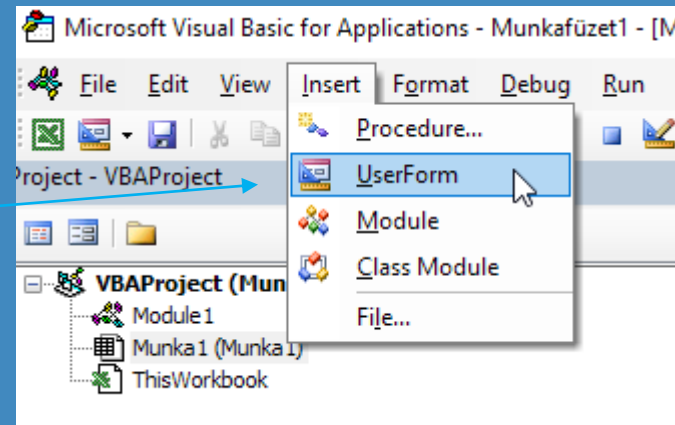
A Windows-os programok mindegyike rendelkezik egy saját ablakkal

Itt találhatók meg a program kezelését szolgáló gombok, menük, stb.

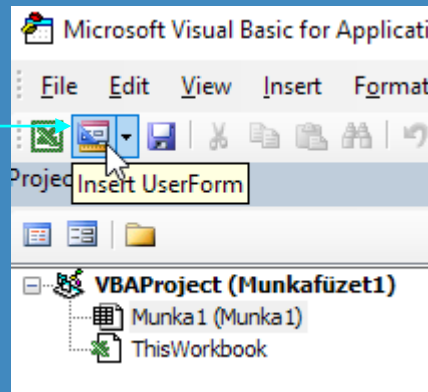
Első feladat, létrehozni ezt az ablakot.

# UserForm

- Menüből:
  - Insert → UserForm

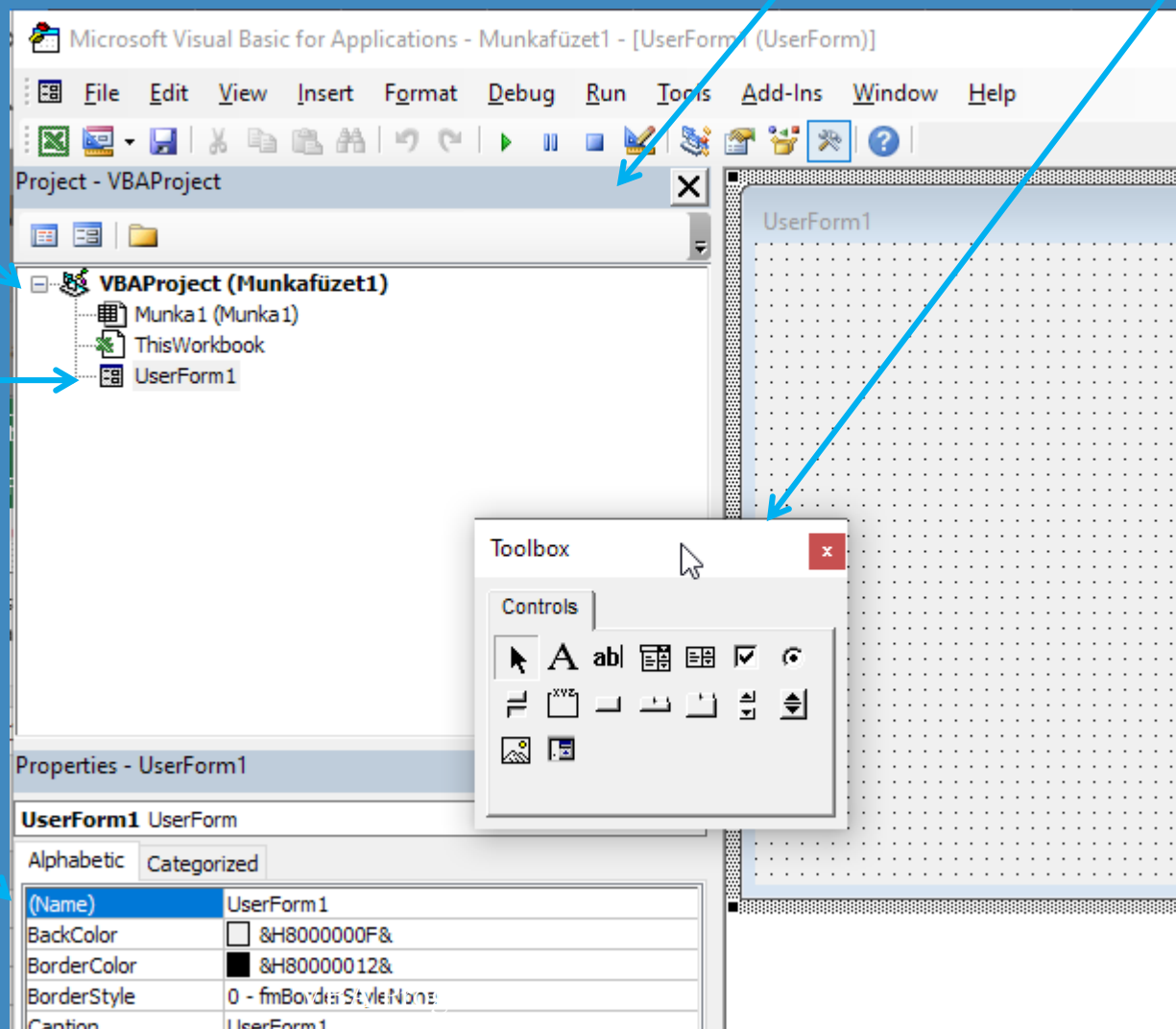


- Ikon-nal





# A fejlesztőrendszer használata



# UserForm

- A UserForm a program (fő)ablaka
- Induló neve: UserForm1
- Nevezzük át...

# Elnevezés

- A vezérlőknek több tulajdonsága (Property) van. Az egyik ezek közül a **Name**. Itt lehet nevet adni nekik. **A név nem tartalmazhat ékezetet!**

Properties - UserForm1	
<b>UserForm1</b> UserForm	
Alphabetic Categorized	
(Name)	UserForm1
BackColor	<input type="checkbox"/> &H8000000F&
BorderColor	<input checked="" type="checkbox"/> &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	UserForm1
Cycle	0 - fmCycleAllForms

Properties - frmProgram1	
<b>frmProgram1</b> UserForm	
Alphabetic Categorized	
(Name)	frmProgram1
BackColor	<input type="checkbox"/> &H8000000F&
BorderColor	<input checked="" type="checkbox"/> &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	UserForm1
Cycle	0 - fmCycleAllForms
DrawBuffer	32000

VBA Programoz

# Elnevezési konvenciók

A név első 3 betűje az elnevezendőről „jön”

- Form  frm
- Mivel ez az első program, legyen a neve: **Program1**
- A teljes neve: **frmProgram1**

# Névadási konvenciók

(Charles Simonyi – Hungarian notation)

## Fontosabb elemek:

- Szövegablak (**TextBox**) txt
- Keret (**Frame**) fra
- Parancsgomb (**CommandButton**) cmd
- Kapcsoló (**CheckBox**) chk
- Opciógomb (**OptionButton**) opt
- Listapanel (**ListBox**) lst
- Üzenetablak (**MessageBox**) msg
- Címke (**Label**) lbl
- Űrlap (**Form**) frm

# Névadási konvenciók

Változónevek esetében a változó típusával kezdünk.

Fontosabb típusok:

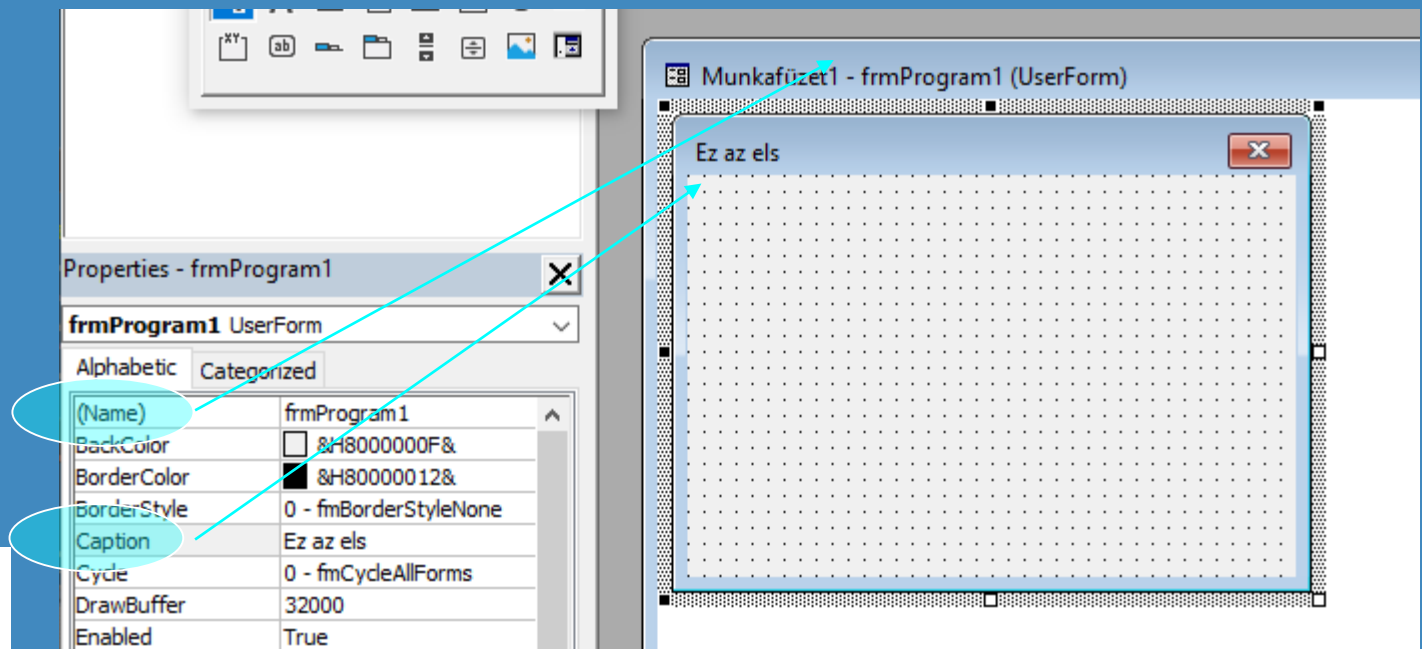
- Egész (**Integer**) `int`
- Valós (**Single**) `sng`
- Dupla pontos valós (**Double**) `dbl`
- Szöveges (**String**) `str`

# Properties (Tulajdonságok)

- Mindig a **kijelölt vezérlőelem** tulajdonságai jelennek meg itt.
- Ezek egy része írható/olvasható, egy része csak olvasható.
- Van ami csak a program készítésekor módosítható.
- Van ami a csak a program futásakor módosítható.
- Van ami csak a program futásakor érhető el.

# Caption

- A vezérlőelem neve: **Name**
- A vezérlőelem felirata: **Caption**
- Nevezzük át ezt is: **”Ez az első programom”**  
(gépelés közben már látszódnia kell)



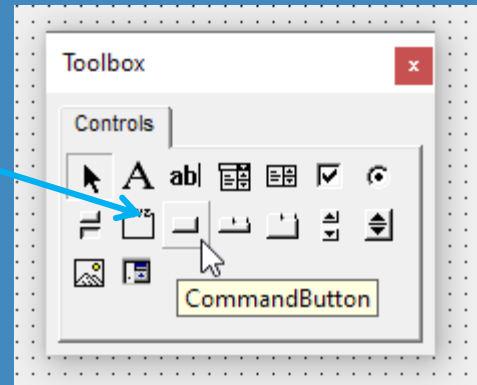


# Parancsgomb (CommandButton)

Tegyünk ki a programot jelképező form-ra egy parancsgombot.

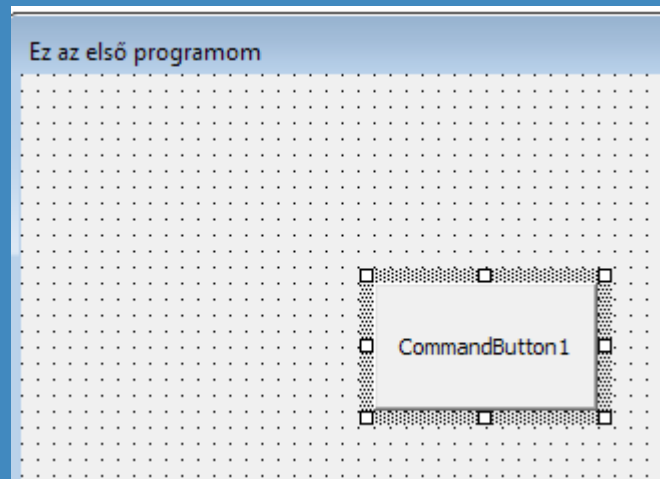
Ha megnyomjuk, íródjon ki egy szöveg.

Keressük meg a Toolbox nevű gyűjteményt, és jelöljük ki a **CommandButtont**



# Parancsgomb (CommandButton)

- Ha sikerült, akkor az egeret a form fölé mozgatva nem egy nyíl, hanem egy szátkereszt jelenik meg.
- A bal egér gomb nyomva tartásával rajzoljunk egy ízléses méretű téglalapot a form-ra.

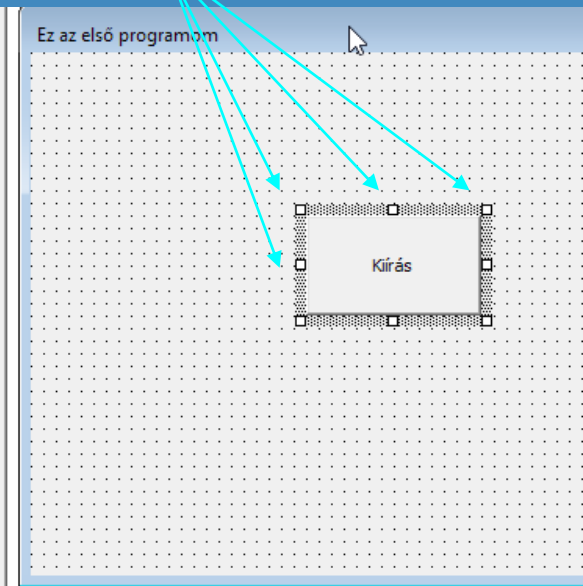
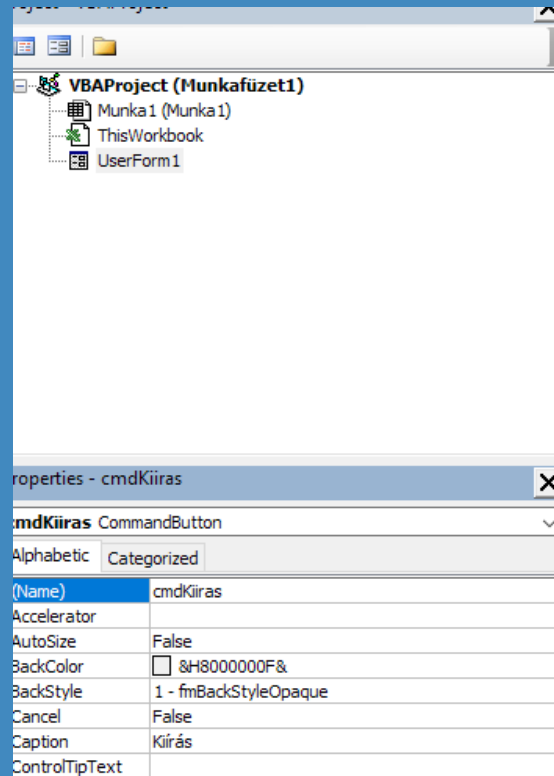


VBA Programozás

# Parancsgomb (CommandButton)

- Neve és felirata automatikusan egyaránt **CommandButton1** lesz.
- Nevezzük át, és adjunk neki feliratot!
- Ehhez ki kell jelölni (a kijelölést a gomb körül látható keret jelzi). A fehér pontok segítségével lehet átméretezni a gombot.

- (Name): **cmdKiir**
- Caption: **Kiírás**



# Esemény (Event)

Mi történik a parancsgomb megnyomásakor?

Egy esemény váltódik ki, és az ahhoz rendelt utasítások végrehajtódnak.

Ennek megadása a következő:

kettőt kell kattintani a Kiírás gombra.

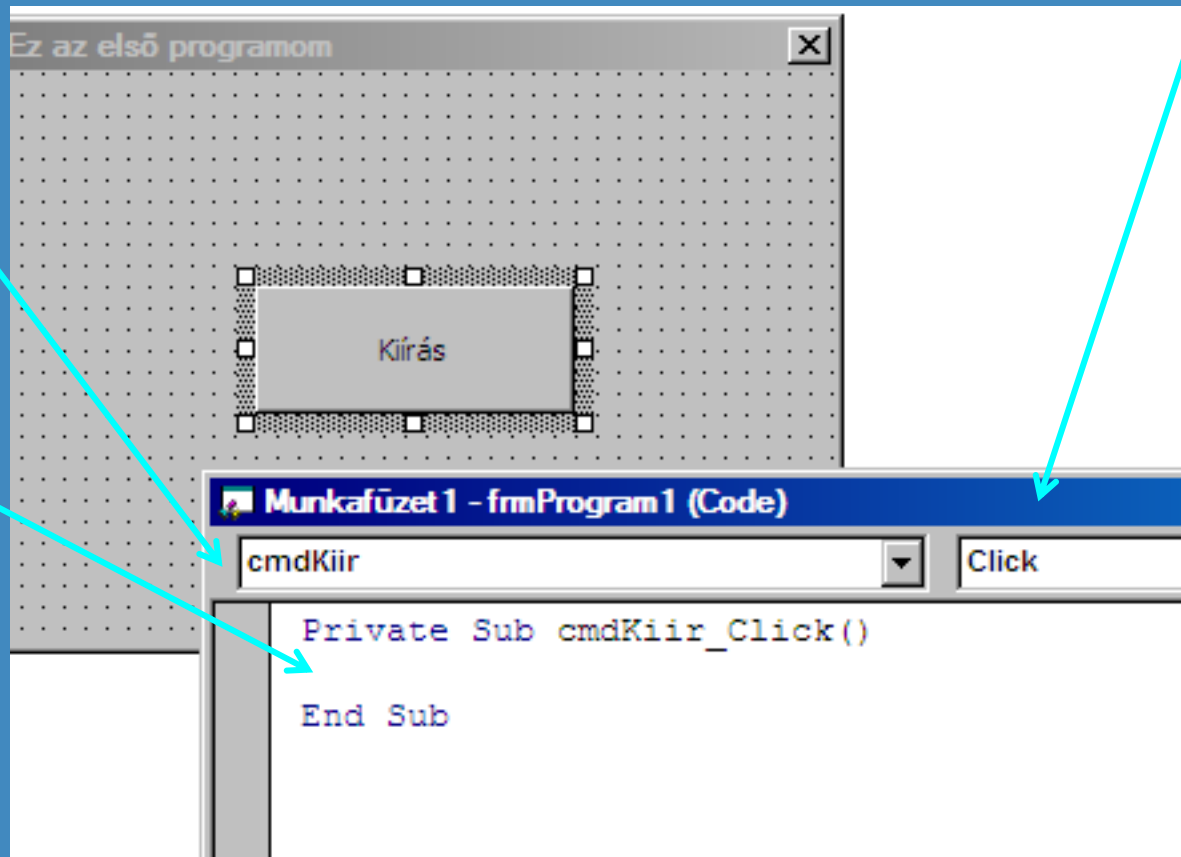
Megjelenik egy új ablak (Code) ahova az utasításokat lehet írni.

# Kód (Code)

A vezérlő neve

A vezérlő  
Click  
eseményéhez  
tartozó kód

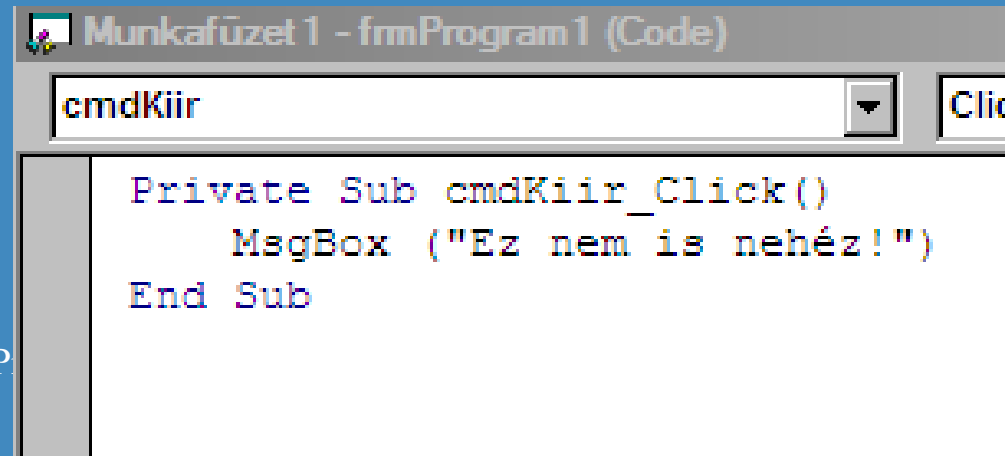
Az esemény



VBA Programozás

# Üzenetablak (MessageBox)

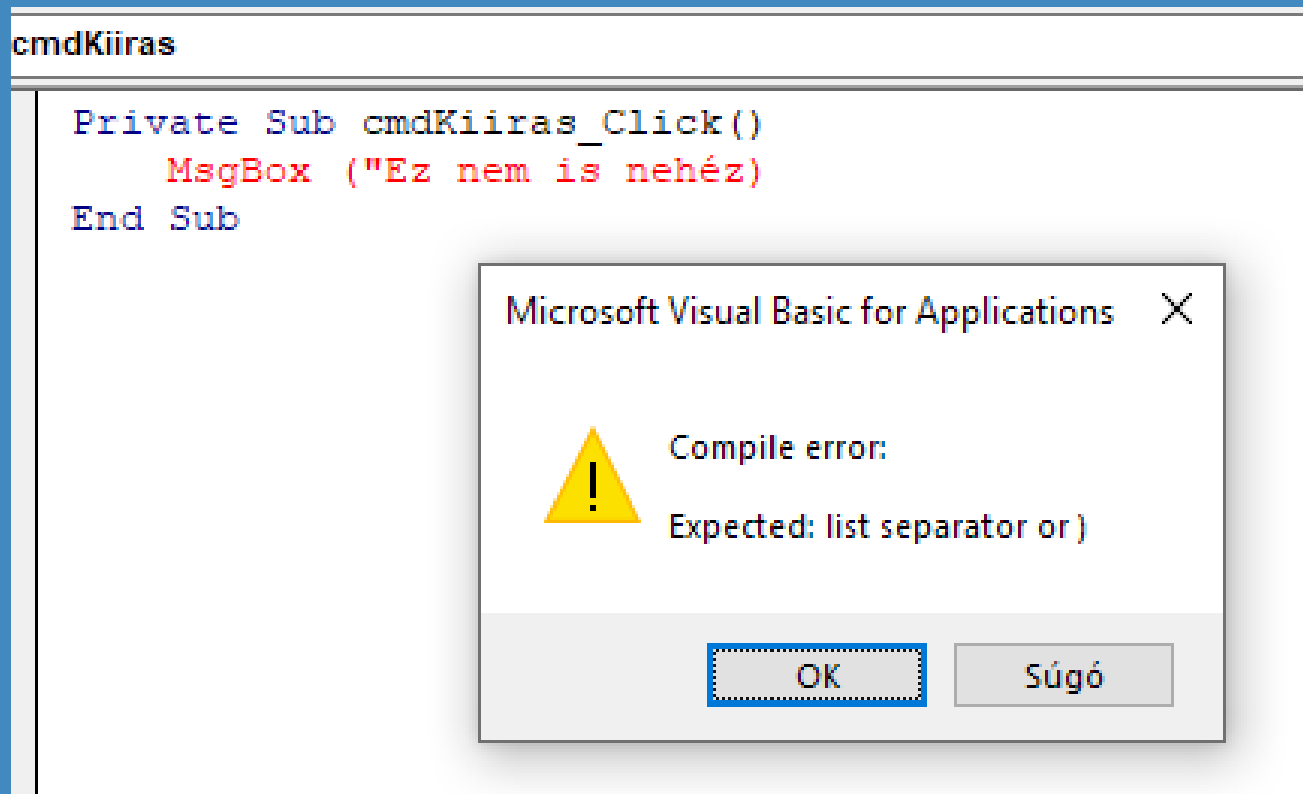
- Legyen a szöveg: „Ez nem is nehéz”
- Üzenetablakba akarjuk kiírni.
- Ennek megadása: **MsgBox(„Ez nem is nehéz”)**
- Lehet nyugodtan kisbetűkkel írni (msgbox), ha nem lett elgépelve, átírja megfelelőre.
- Ha begépelés után piros lesz a sor, akkor valami elgépelésre került. Tessék korrigálni...



```
Private Sub cmdKiir_Click()  
    MsgBox ("Ez nem is nehéz!")  
End Sub
```

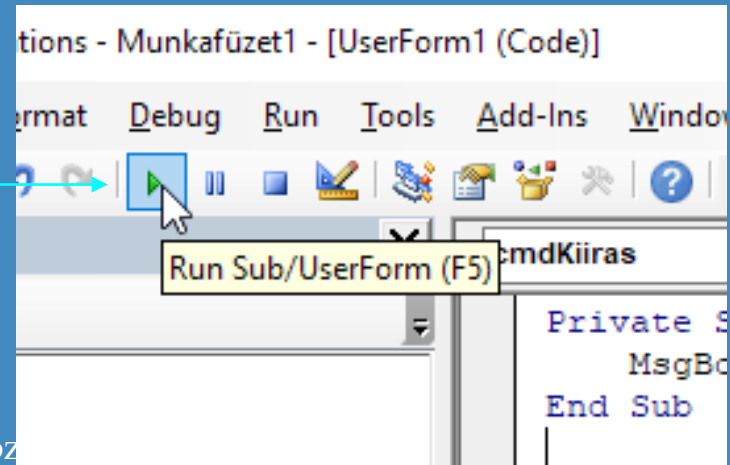
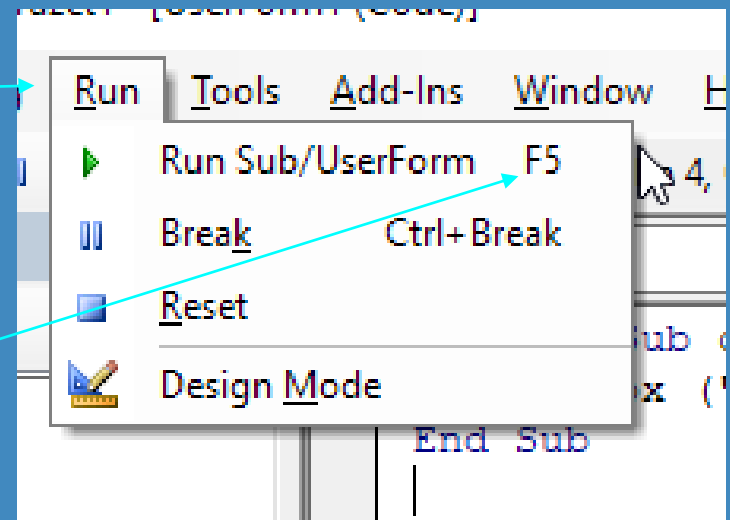
VBA P

# Hiba...



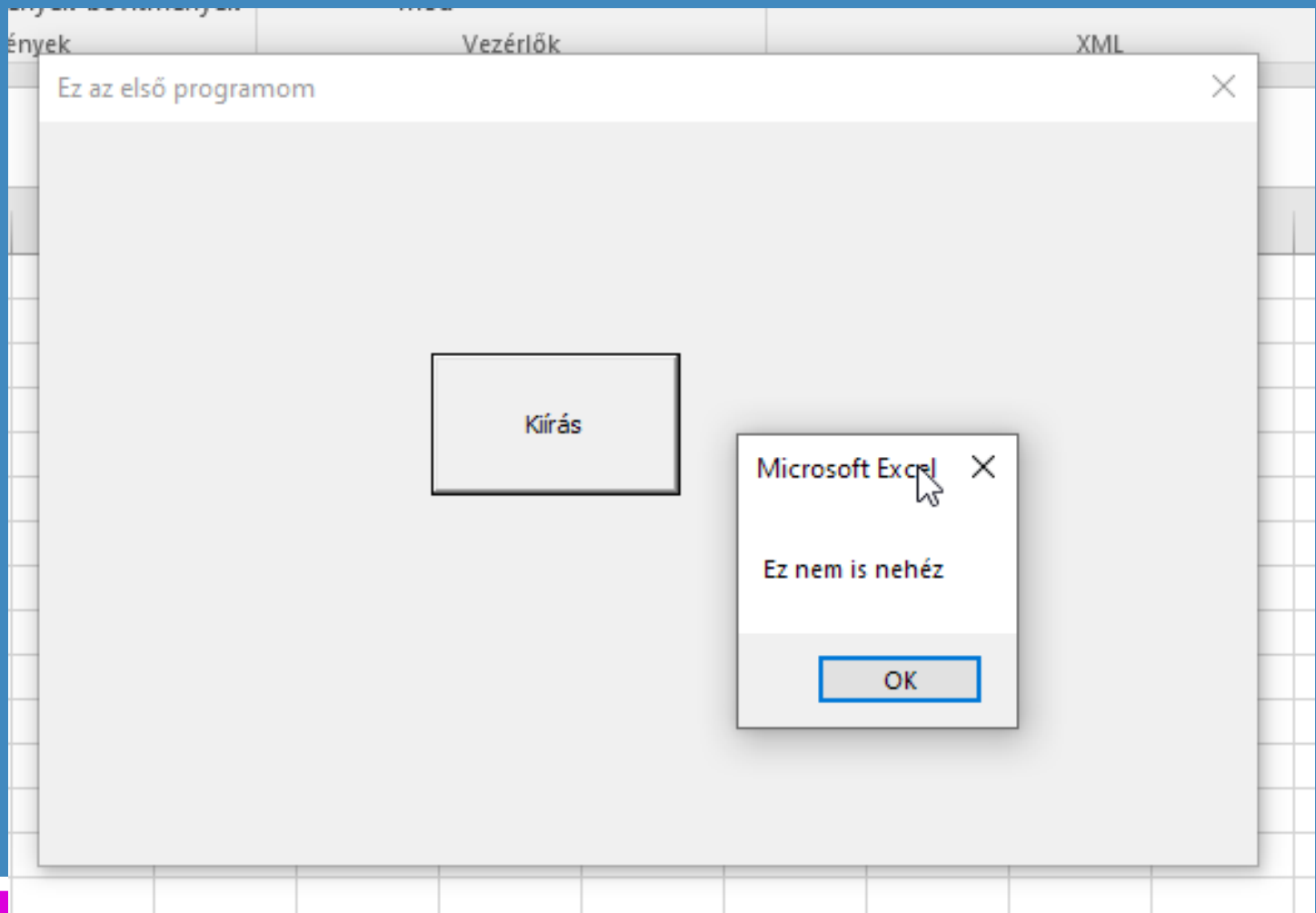
# Futtatás (Run)

- Menüből:
- Funkció gombbal: **F5**
- Ikon soron levő gombbal:



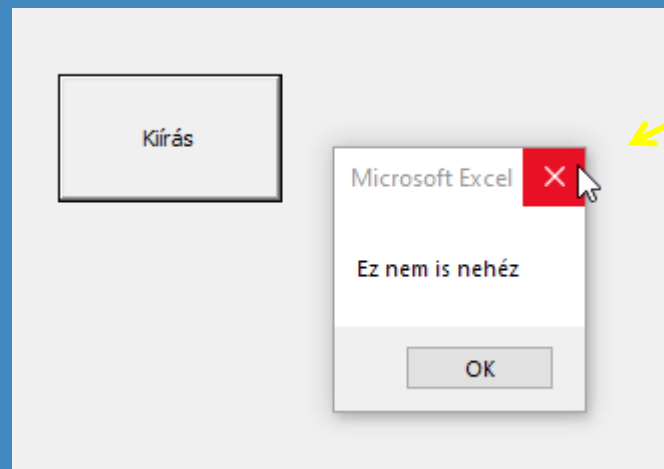


# Eredmény



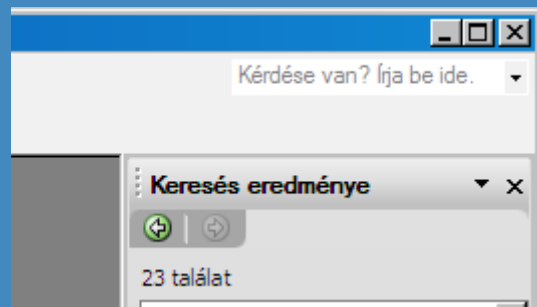
# Kilépés a programból (Exit)

- Nem készítettünk rá gombot, tehát egyszerűen:



# Kilépés a VBA-E-ből

- File → Close and Return Microsoft Excel
- Alt Q



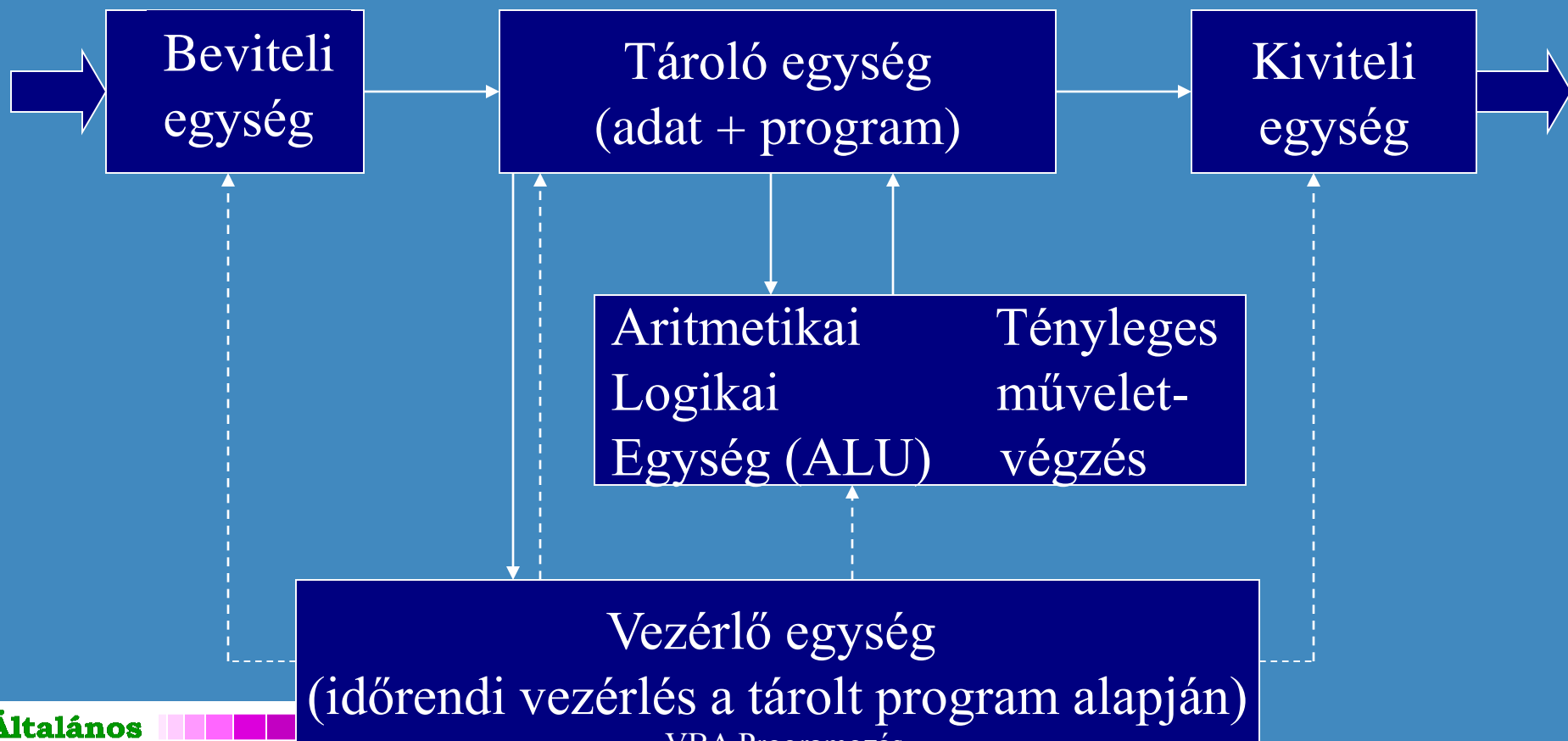
# Mentés

- A VBA-E-ből kilépve visszajutottunk az Excel táblázatához.
- A program mentése a Munkafüzet mentésével történik (annak ellenére, hogy az most üres).
- Legyen a név: Prg1.xlsm (makróbarát)

# Mentett VBA program használata

- Kilépni az Excel-ből, majd
- Megkeresni a mentett **Prg1.xlsm** fájlt.
- Kettős kattintással elindítani.
- A program látszólag nincs sehol...
- ...elindítani a VBA-E-t
- A baloldali **Project – VBAProject** ablakban a forms-ok között megkeresni az **frmProgram1**-et
- Kettős kattintás rajta, és megjelenik a form.

# A számítógép klasszikus funkcionális rendszervázlata



VBA Programozás

# Meghatározás

- Számítógép: olyan technikai rendszer, amely adatok, információk feldolgozására képes, közvetlen emberi beavatkozás nélkül a benne letárolt utasítások alapján.
- Erőforrás: a rendeltetésszerű használathoz szükséges komponensek összessége.

# Erőforrások (1)

## Hardware

A számítógép fizikai megvalósítása

1. Központi egység (CPU)
2. Központi memória (adatok és utasítások tárolására)
3. Áramkörök az adattovábbításra (sin, busz,...)

## Software

Programok, utasítások összessége

1. Rendszerszoftverek (a szgéppel együtt megvásárolhatók, a felhasználó munkáját könnyítik)
  - a) Vezérlő szoftver (operációs rendszer, amely az erőforrások optimális kihasználtságát maximalizálja)
  - b) Feldolgozó szoftver (szövegszerk., segédprg., ...)

ogra



# Erőforrások (2)

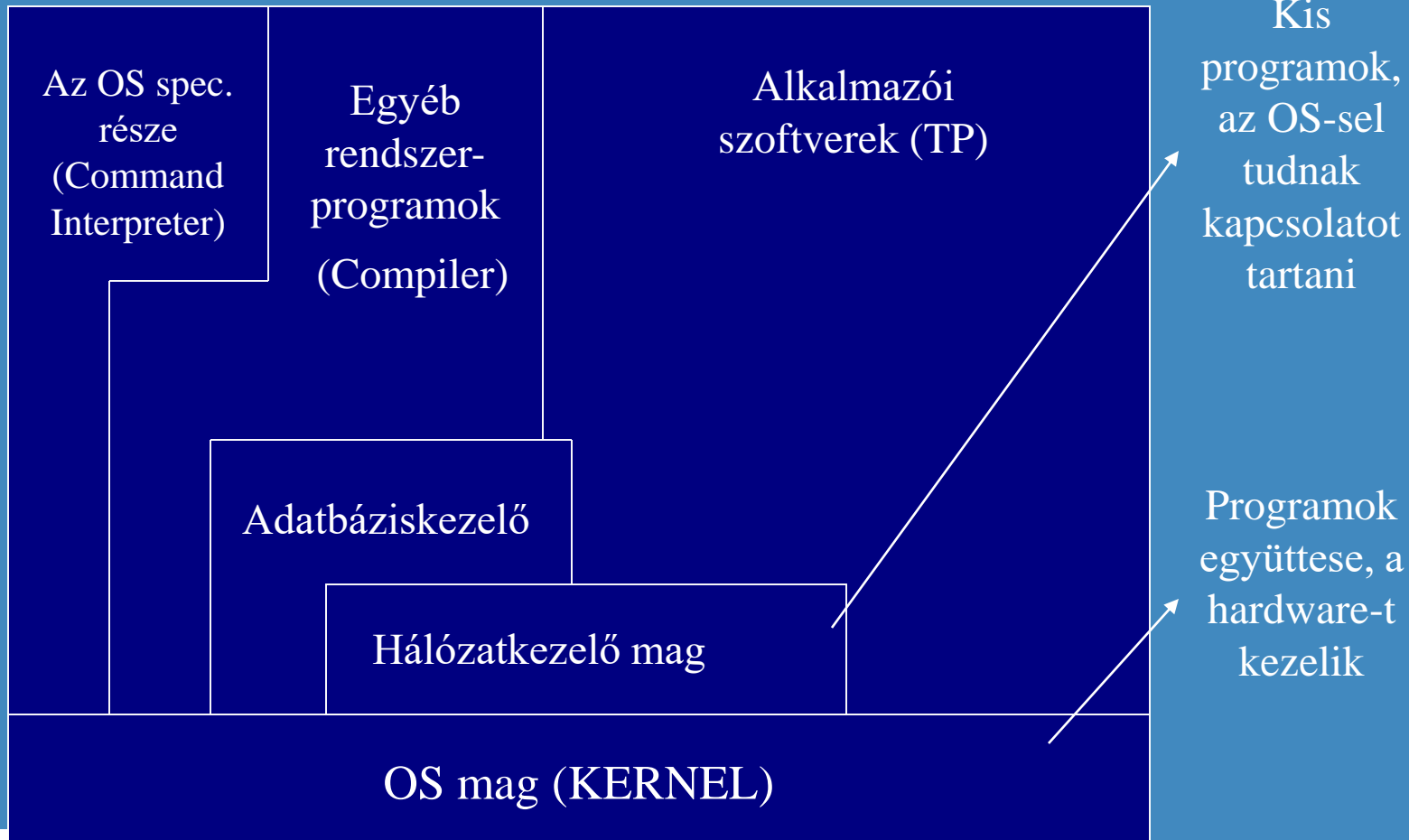
## 4. Perifériák:

- a). Tömegtárolók (mágneslemez,.)
- b). Kapcsolattartás a felhasználóval (keyboard, display, mouse, tablet, plotter, printer,...)
- c). Más rendszerekkel való kapcsolattartás eszközei (hálózati kártya, modem, ...)

## 2. Alkalmazói szoftverek (egyedi célra írottak, pl.: Word, Photoshop, ...)

# Erőforrások (3)

## Felhasználó



# A számítógépes feladatmegoldás eszközei

## Adatok

(Amiken utasításokat  
hajtunk végre)

## Utasítások

(Amiket végrehajtunk)

Program struktúra

# Adatok

- Konstans (a programon belül végig állandó)
- Változó (a program során más és más értéket vehet fel)
- Adattípusok (felvehető értékük szerint)
  - numerikus
    - egész
    - valós
  - szöveges
    - karakter
    - sztring
  - logikai

# Algoritmus (definíció)

Egy meghatározott cél elérésére irányuló, egymástól elkülönített tevékenységek alkalmas sorozata, melynek segítségével bizonyos kiindulási helyzetből, végesszámú közbenső állapoton keresztül, előírt feltételeknek eleget tevő véghelyezethez jutunk.

# Példa

## egyszerű numerikus algoritmusra

$$c = a + b$$

- vedd a-t
- vedd b-t
- képezd  $a+b$  -t
- legyen  $c = a+b$  -vel
- tedd c-t
- vége

# Numerikus algoritmus jellemzői

- Diszkrét jellegű: Véges sok elemi lépés. (A leírása!!)
- Determinált (meghatározott): Mindig tudjuk, a következő lépésben mi lesz a teendő.
- „Elemiek” a lépések: Már nem függ külső paraméterektől.
- Célra irányított (teljes): Van egy értékrendszer, ami ezt a célt kifejezi. Ezt akarjuk elérni.
- Véges: Véges számú lépésben eléri célját. (A megoldása!)
- Egész feladatosztályra érvényes: A bemenő adatok értelmezési tartománya: minden lehetséges adat.

# Algoritmus leírási módok (1)

- természetes nyelv (vagy más néven verbális.) Gyors, mindenki által könnyen érthető, de pontatlan.

Pl.: kerékcseré:

Labilis mozgást érzékelek; jelzek jobbra; leállok; kulcs kikapcs; kézifék be.

Amíg nem szállhatok ki, hátranézek.

Kiszállok, ..., szükséges szerszámokat előveszem;

Dísz tárcsa le, rögzítőcsavart lazít, ..., emelek, kereket le.

Ha van pótkerék, akkor:

kiveszem pótkereket, kereket be, pótkereket fel, ..., légnyomást ellenőriz. Ha megfelelő semmit, különben

még meg nem felel  
pumpál, ellenőriz....

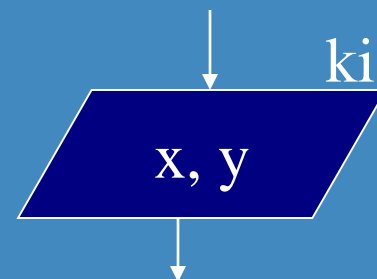
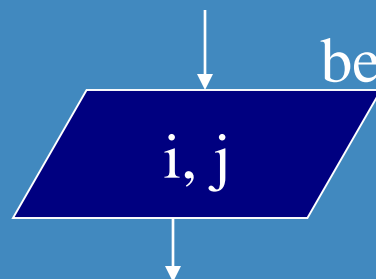
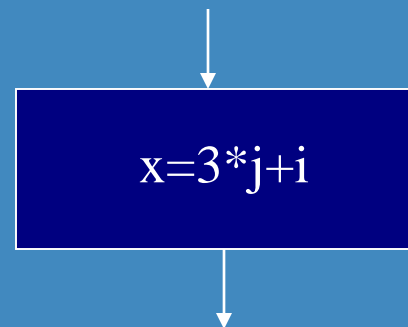
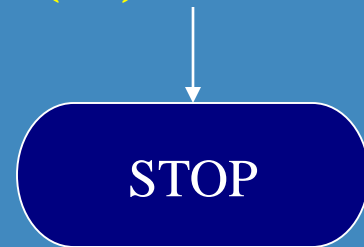
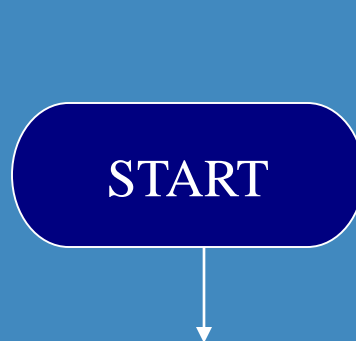
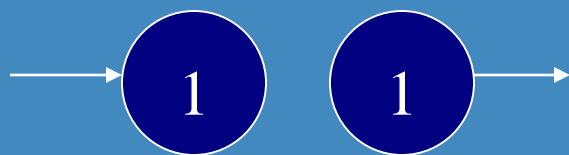
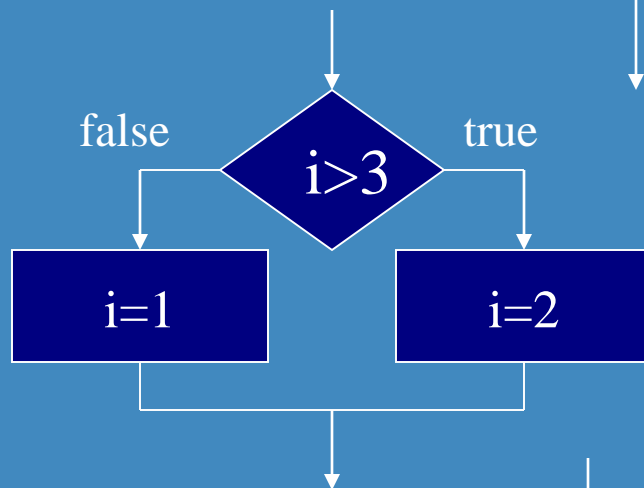
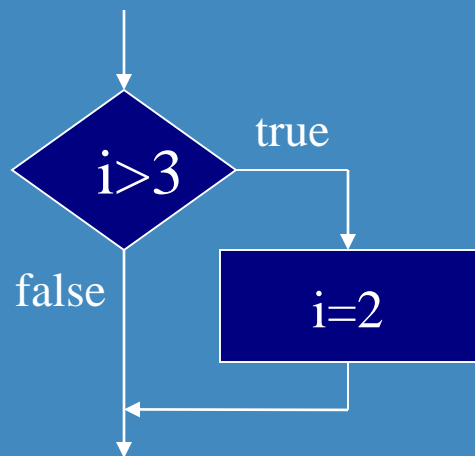
Ha nincs, ragaszt, majd pumpál, stb.. , majd elindul

VBA Programozás



# Algoritmus leírási módok (2)

- folyamatábra: többféle elterjedt jelölésrendszer létezik.



# Algoritmus leírási módok (3)

Pszeudonyelv: elemi utasítások összessége. Olyan vezérlő vagy keret utasítások, amelyek az algoritmus logikai vázát adják. (Pszeudo azért, mert nem programnyelv, hanem viszonylag kötetlen, programnyelvszerű jelölésrendszer. Gyakran emlegetik **PDL**, azaz **P**roblem **D**efinition **L**anguage - probléma leíró nyelv - néven is.)

Pl.:      program átvált  
         olvas dollár  
         forint = dollár \* árfolyam  
         kiír forint  
         vége átvált

# Algoritmus leírási módok (4)

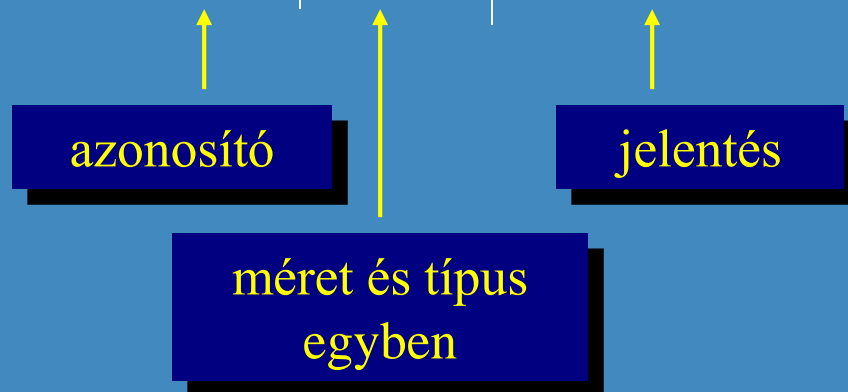
- Hiánya: kódoláskor nem derül ki az adatokról semmi (méret, cél, típus), ezért a PDL leíráshoz mindig hozzátartozik egy **ADATLEÍRÁS**.

Felépítése:

- azonosító
- méret
- típus
- jelentés

- Pl.:  $\text{spoz} = \sum x_i \ (i=1\dots n)$ , ha  $x_i > 0$

n	integer	az elemek száma
spoz	real	a pozitív elemek összege
x(n)	real	az adatok tömbje
i	integer	ciklusváltozó



# Algoritmus leírási módok (5)

Metanyelv (a „nyelv nyelve”): Pl.:

Sajátos szókészlettel és nyelvtannal rendelkező nyelv, melynek segítségével egy nyelv formai és tartalmi elemzése elvégezhető.

Backus-Naur Forma (BNF)

A BNF szimbólumrendszer a PASCAL nyelv leírásához való

Szimbólumok:













$::=$  szabály (definíció szerint)

| választás (vagy)

{...} ismétlés nullaszer vagy többször

$\langle \dots \rangle$  szintaktikai egység

# Algoritmus leírási módok (6)

 <b>TIOBE</b> <small>(the software quality company)</small>					<a href="#">Schedule</a>
Sep 2023	Sep 2022	Change	Programming Language		Ratings
1	1			Python	14.16%
2	2			C	11.27%
3	4	▲		C++	10.65%
4	3	▼		Java	9.49%
5	5			C#	7.31%
6	7	▲		JavaScript	3.30%
7	6	▼		Visual Basic	2.22%
8	10	▲		PHP	1.55%
9	8	▼		Assembly language	1.53%
10	9	▼		SQL	1.44%
11	15	▲		Fortran	1.28%

Bármely magasszintű  
programozási nyelv:  
VBA, Python,  
JavaScript, SQL, C,  
C++, C#, Java, PHP, R,  
stb...

<https://www.tiobe.com/tiobe-index/>

# A program

## Számítógépi program:

- kezelt adatok leírása
- elvégzendő tevékenységek

## Programozási nyelv:

szimbólumrendszer, melynek segítségével programot írhatunk.

## Progr. nyelv fő összetevői:

- karakterkészlet
  - nyelvi elemek
    - kulcsszavak, operátorok
    - azonosítók, konstansok, utasítások
  - szintaktika (nyelvtan)
  - szemantika (jelentéstan)
- (A repülőgép ezután hajszárait tépdesve kockásan ráugrott a levegőre.)**

# Azonosítók

Mire: konstansok, típusok, változók, függvények, szubrutinok jelölésére.

Szintaxisuk: betűvel kezdődő alfanumerikus karaktersorozat.  
Max 254 karakter. A névben lehet aláhúzásjel \_  
(pl: Fizetes\_Osszesen)

Ékezetes karakter, és helyköz névben ne legyen!

Hatáskörük: használatuk hatáskörükön belül egyértelmű és kizárólagos. (később pontosítjuk)

A kis és nagy betűk NEM különböztetődnek meg!

(pl.: x, y, x2, valtozo, Valtozo, var5, Osszeg, ...)

Standard szubrutin, függvény azonosítók

(pl.: MsgBox, InputBox, exp, cos, acos, fopen, fgetc,...)

Fontos! Vannak foglalt azonosítók

→ kulcsszavak pl.: Dim, For, Next, If, Integer, ...

# Megjegyzés (komment)

- ' Ez egy megjegyzés, és nem lehet több soros.
- ' Ha több soros kell, akkor minden sor
- ' elejére ki kell tenni

## Tanácsok:

- Célszerű minden szubrutin, függvény elején megadni mit csinál
- Célszerű a módosításokat itt vezetni (mikor mi módosult)
- Célszerű minden változóról nyilatkozni, mi a szerepe
- Célszerű a program írása közben vezetni, nem utólag beírni



# VBA program szerkezete (1)

- Egy VBA program szubrutinokból, és függvényekből áll
- A függvények meghívhatják egymást
- Meghívhatják önmagukat is (rekurzió)
- Célszerű mindent definiálni illetve deklarálni

# Konstans definíció

- Azonosítót rendelünk konstans értékhez, kifejezéshez. Ez az azonosító - a konstans neve - szolgáltatja a konstans értéket.

A program futása alatt nem változtatható!

# Konstans lehet

- Szám:
  - decimális egészek (int): 325, +325, -400, 0 (nem nullával kezdődik)
  - valósok (real):
    - fixpontos: 0.1, +0.1, -0.012
    - lebegőpontos: 0.1E0, +0.1e-1, 0.0012E+1
- Karakter: `a` `1` `ab` (aposztrófok közé írt 1 vagy több karakter)  
a több karakteres konstans nem szabványos!!
- Karakterlánc (sztringkonstans): (idézőjelek közé írt karakterek)  
”ez egy meglehetősen hosszú sztring” ”ez rövid”
- Const Szam1 As Integer = 4
- Const Szam2 = 0.321, Szam3 = 12
- Const Nev As String = ”Kérem adja meg a nevét:”

# Utasítás

- Függvényhívó utasítás: (pl.: MsgBox(), InputBox(),...)
- Értékadó utasítás: (az a tevékenység, melynek során egy változó értékét módosítjuk)  
 $\langle \text{változó} \rangle = \langle \text{kifejezés} \rangle$

kifejezés: (konstans vagy változó) operandusokból és operátorokból álló kiértékelhető szerkezet.

(pl.:  $a = 30.0 * 3.1415 / 180.0$ )

# Változó deklaráció

- A memória egy része, ahol a program futása közben más - és más (változó) értékek jelenhetnek meg.

Minden változónak van:

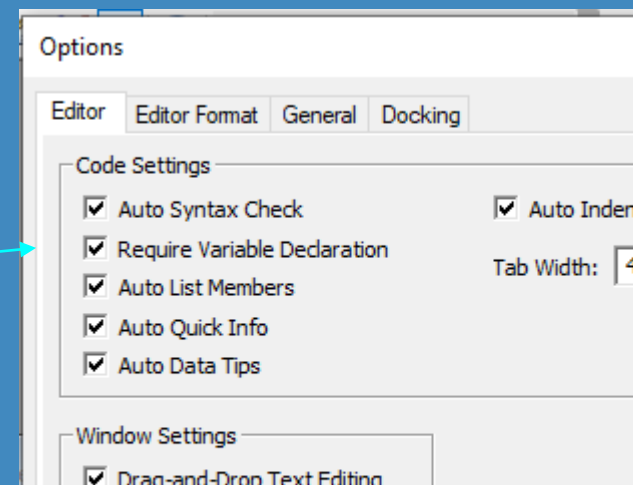
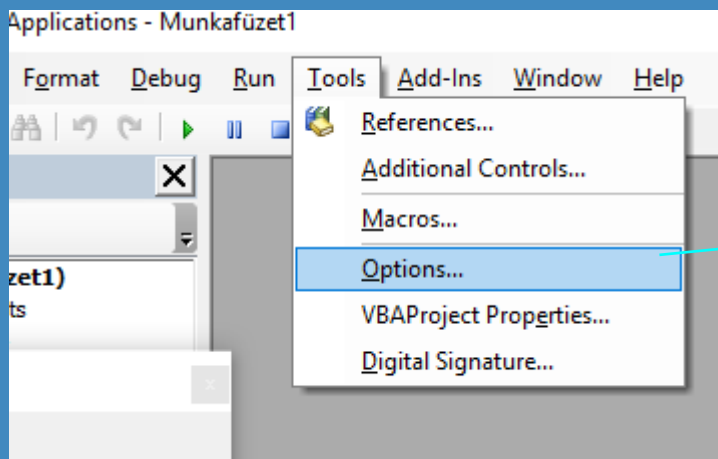
- Neve: a tárolóhely címére utal. Ez a cím egyelőre elérhetetlen.
- Címe: a memória területet azonosítja.
- Típusa: a névvel címzett memóriarész értelmezése (a programozó szemszögéből!)
- Aktuális értéke (tartalma): a névvel címzett memóriarész (tároló) tartalma egy adott időpillanatban.
- Érvényességi köre: a program azon része, amelyben a változóra a nevével érvényesen hivatkozni lehet. (Dim, Static, Private, Public)

# Fontosabb változó típusok

Név	Méret	Értéktartomány
Byte	1 Byte	0 ... 255
Boolean	2 Byte	True vagy False
Integer	2 Byte	-32 768 ... +32 767
Long	4 Byte	-2 147 483 648 ... +2 147 483 647
Single	4 Byte	-3.402823E38 ... -1.401298E-45 +1.401298E-45 ... + 3.402823E38
Double	8 Byte	-1.79769313486232E308 ... -4.94065645841247E-324 +4.94065645841247E-324 ... +1.79769313486232E308
Date	8 Byte	0100 január 1 ... 9999 december 31
String (változó)	10 + String	0 ... kb. 2 milliárd karakter
String (fix)	String	1 ... kb. 65 400 karakter

# Változók deklarációja

- Célszerű minden változót deklarálni
  - A program gyorsabban fut
  - Kevesebb memóriát használ
  - Csökken a név elgépelésének az esélye
- Kikényszerítése:
  - A modul elején a General részben: **Option Explicit**
  - **Vagy:**

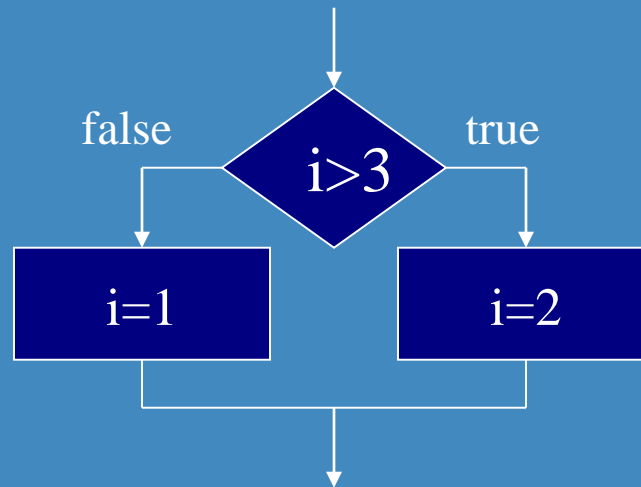
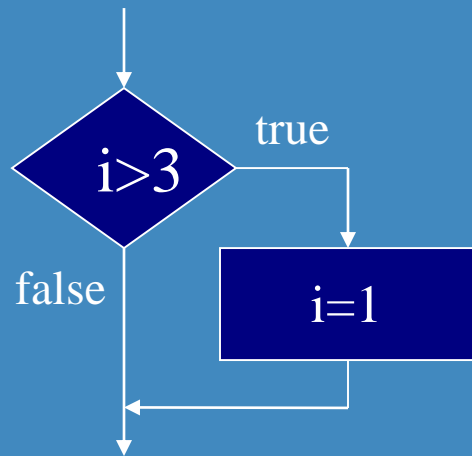


# Változók deklarációja

- Például:
  - Dim x As Integer
  - Dim y As Long
  - Dim Nev1 As String, Nev2 As String \* 50
- Tipikus hiba:
  - Dim i, j, k As Integer
  - (A fenti esetben i és j Variant lesz, k pedig Integer)
- Helyesen:
  - Dim i As Integer, j As Integer, k As Integer



# Vezérlési szerkezetek (1)



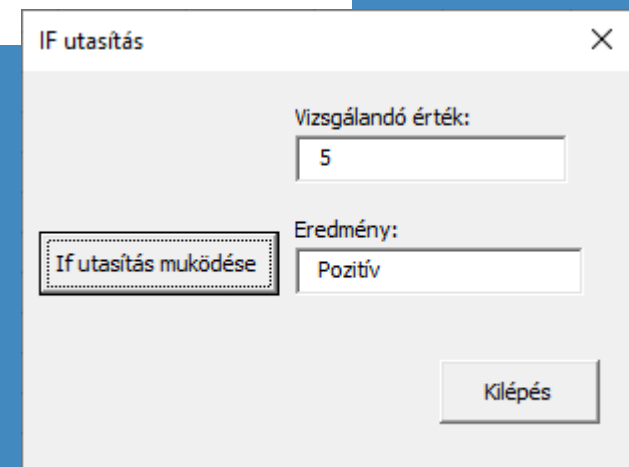
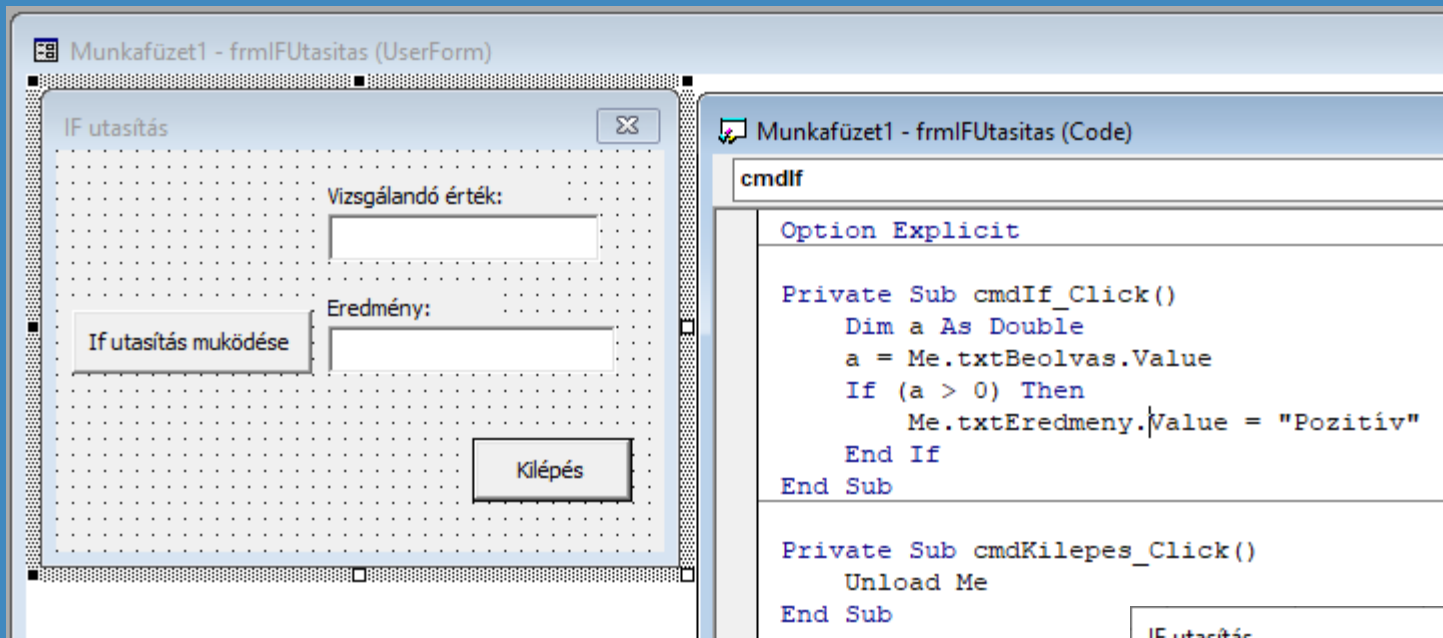
Működése:

**If (logikif) Then ... End If** szerkezet esetén ha az **If** utáni logikai kifejezés értéke igaz, akkor a **Then** utáni utasítás végrehajtódik. Egyébként az **End If** UTÁNI utasítás hajtódik végre.

**If (logikif) Then ... Else ... End If** szerkezet esetén a logikai kifejezés hamis értékénél nem az **End If**, hanem az **Else** utáni utasítás hajtódik végre.

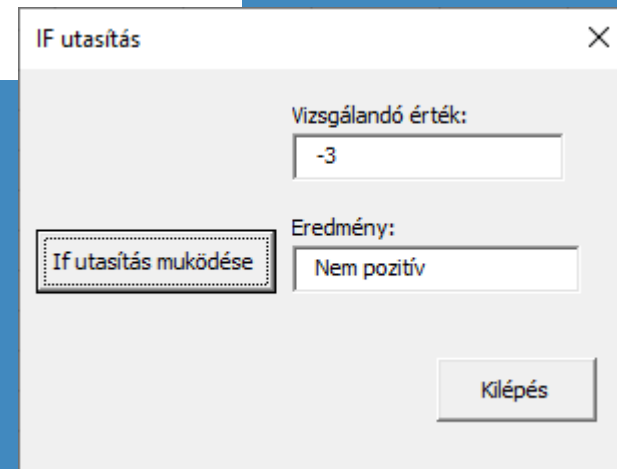
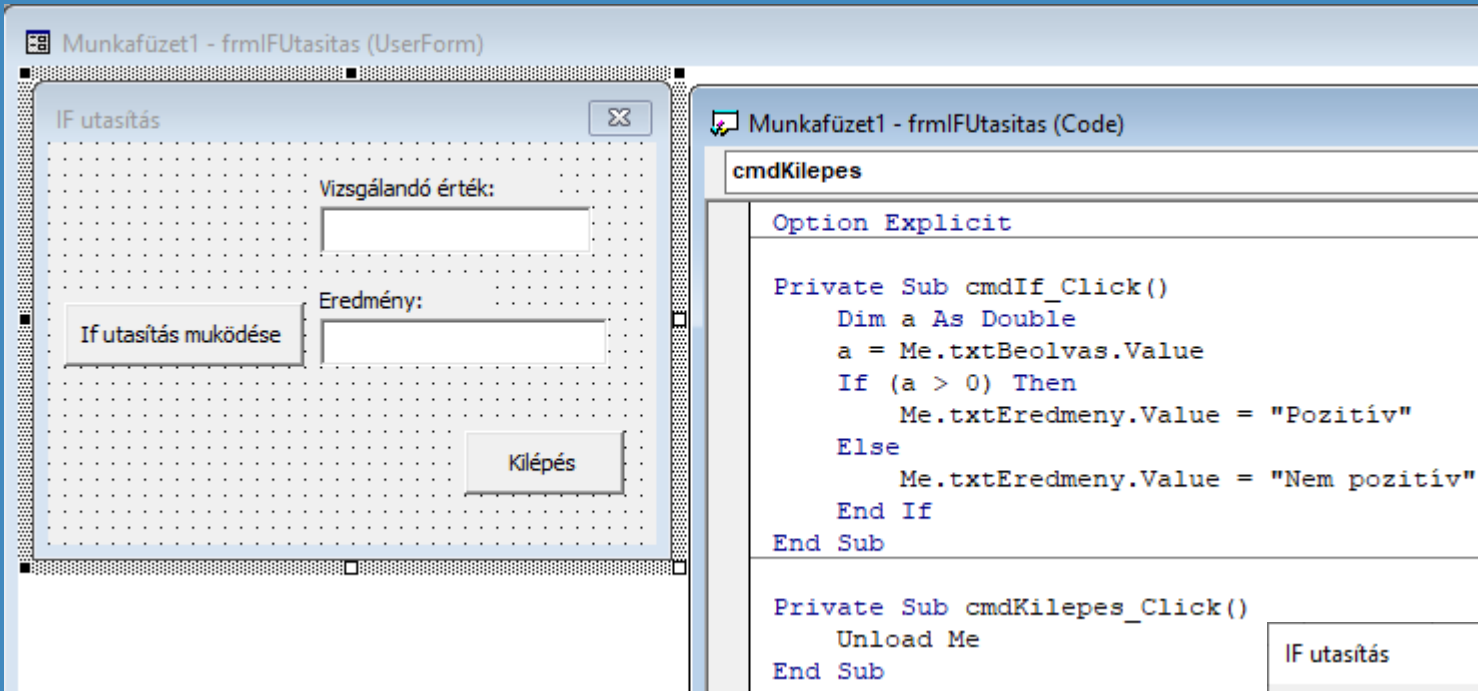
# Vezérlési szerkezetek (2)

If (logikif) Then ... End If utasítás példa:



# Vezérlési szerkezetek (3)

If (logikif) Then ... Else ... End If utasítás példa:



# Vezérlési szerkezetek (4)

If (logkif) Then ... ElseIf (logkif) Then... Else ... End If utasítás példa:

IF utasítás

Vizsgálandó érték:

Eredmény:

If utasítás működése

Kilépés

Munkafüzet1 - frmIFutasitas (Code)

```
cmdIf

Option Explicit

Private Sub cmdIf_Click()
    Dim a As Double
    a = Me.txtBeolvas.Value
    If (a > 0) Then
        Me.txtEredmeny.Value = "Pozitív"
    ElseIf (a = 0) Then
        Me.txtEredmeny.Value = "Nulla"
    Else
        Me.txtEredmeny.Value = "Negatív"
    End If
End Sub

Private Sub cmdKilepes_Click()
    Unload Me
End Sub
```

IF utasítás

Vizsgálandó érték:

Eredmény:

If utasítás működése

Kilépés

IF utasítás

Vizsgálandó érték:

Eredmény:

If utasítás működése

Kilépés

IF utasítás

Vizsgálandó érték:

Eredmény:

If utasítás működése

Kilépés

# Logikai kifejezés (1)

Mik képeznek logikai kifejezést?

- Relációs operátorok bármilyen kifejezésekkel  
(  $a \geq 5$  )
- logikai operátorok logikai kifejezésekkel.

Logikai értéket képviselhetnek:

- Konstansok (pl.: `Const Igaz As Boolean = True`)
- Változók: (pl.: `Dim Logikai As Boolean`)

Ezek kiértékeléskor logikai értéket eredményezhetnek.

# Logikai kifejezés (2)

IF utasítás

If utasítás működése

Eredmény:

Kilépés

Munkafüzet1 - frmIFutasitas (Code)

cmdKilepes

```
Option Explicit

Private Sub cmdIf_Click()
    Const Igaz As Boolean = True
    Const Hamis As Boolean = False
    If (Igaz) Then
        Me.txtEredmeny.Value = "Igaz"
    End If
End Sub

Private Sub cmdKilepes_Click()
    Unload Me
End Sub
```

IF utasítás

If utasítás működése

Eredmény: Igaz

Kilépés

IF utasítás

If utasítás működése

Eredmény:

Kilépés

Munkafüzet1 - frmIFutasitas (Code)

cmdKilepes

```
Option Explicit

Private Sub cmdIf_Click()
    Const Igaz As Boolean = True
    Const Hamis As Boolean = False
    If (Hamis) Then
        Me.txtEredmeny.Value = "Igaz"
    Else
        Me.txtEredmeny.Value = "Hamis"
    End If
End Sub

Private Sub cmdKilepes_Click()
    Unload Me
End Sub
```

IF utasítás

If utasítás működése

Eredmény: Hamis

Kilépés

# Relációs operátorok

= < > <= >= <>

Precedenciájuk az aritmetikai operátorok után.

# Precedencia

(A műveletek végrehajtásának sorrendje)

## Alapszabályok:

1. Zárójelek közötti kifejezések kiértékelése, mintha a zárójelen belül egy operandus lenne. Mindig a legbelső zárójelen belül kezdődik meg a kiértékelés.

$$\text{Pl.: } (x + 1) < ( (x - \text{abs}(x)) * 1.6)$$

2. Azonos precedenciájú operátorok kiértékelése balról-jobbra történik. (Általában, bár a compiler optimalizációs célból átrendezheti a kiszámítást.)

$$\text{Pl.: } 7.0 + a - 1.0$$

3. Különböző precedenciájú operátorok esetén a magasabb precedenciájú értékelődik ki előbb. (Precedencia táblázat!)

$$\text{Pl.: } 7 + a * 9 / \text{abs}(i)$$



# Precedencia táblázat

- Hatványozás (^)
- Egy operandusú (+, −)
- Szorzás, osztás (\*, /)
- Egész osztás (\)
- Maradékképzés (Mod)
- Összeadás, kivonás (+, −)
- String összekapcsolás (&)
- Aritmetikai bit műveletek (<<, >>)
- Relációs operátorok (=, <>, <, <=, >, >=)
- Logikai negáció (Not)
- Logikai szorzás (And)
- Logikai összeadás (Or)

# Logikai operátorok

**AND:** (ÉS, logikai szorzás)

**OR:** (VAGY, logikai összeadás)

**NOT:** (NEM, logikai negáció)

A logikai operátorokat háromféleképpen szokás tárgyalni:

- igazságtáblával
- kapcsolóáramkörrel
- Venn-diagrammal (halmazsal)

# Igazságtáblával

AND	F	T
F	F	F
T	F	T

OR	F	T
F	F	T
T	T	T

NOT	F	T
	T	F

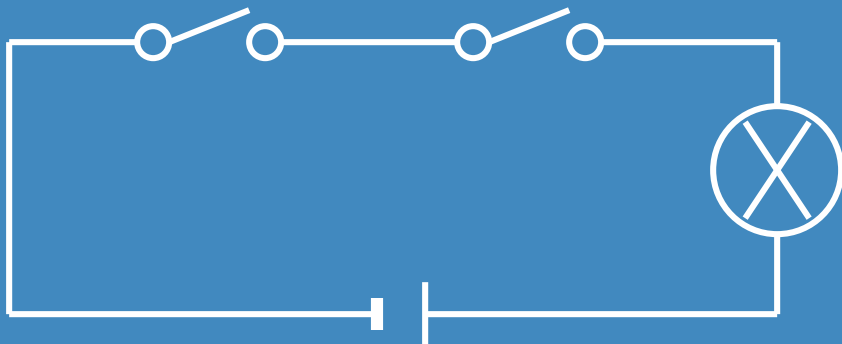
Vagy:

AND		
F	F	F
F	T	F
T	F	F
T	T	T

OR		
F	F	F
F	T	T
T	F	T
T	T	T

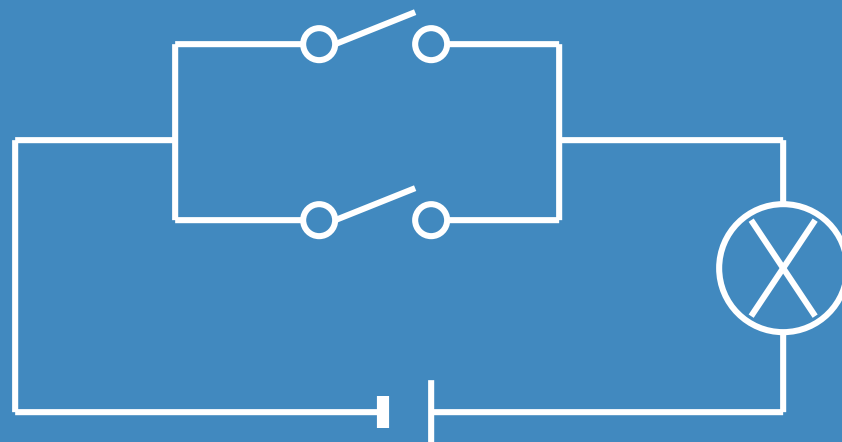
NOT	
F	T
T	F

# Kapcsolóáramkörrel



**AND**

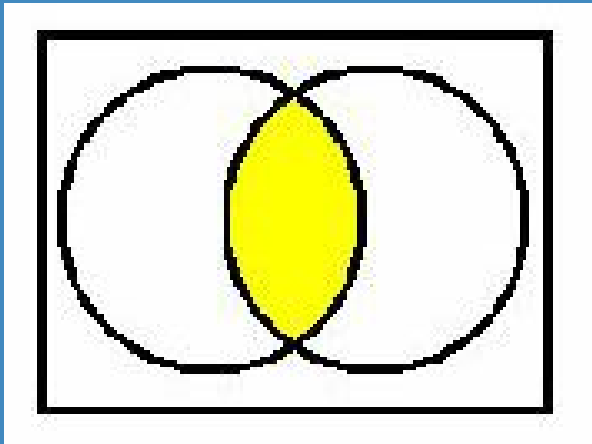
A lámpa ég, ha  
**mindkettő** be van  
kapcsolva



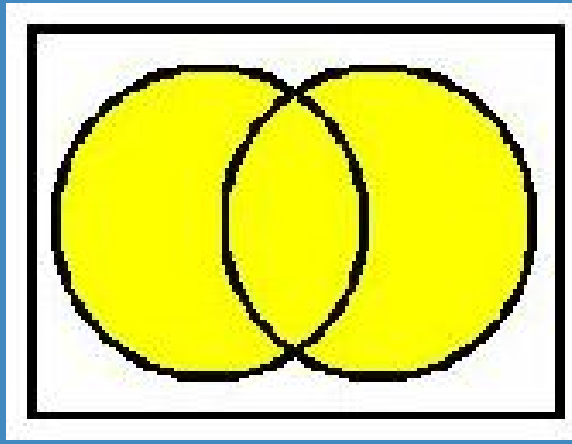
**OR**

A lámpa ég, ha  
**legalább az egyik** be  
van kapcsolva

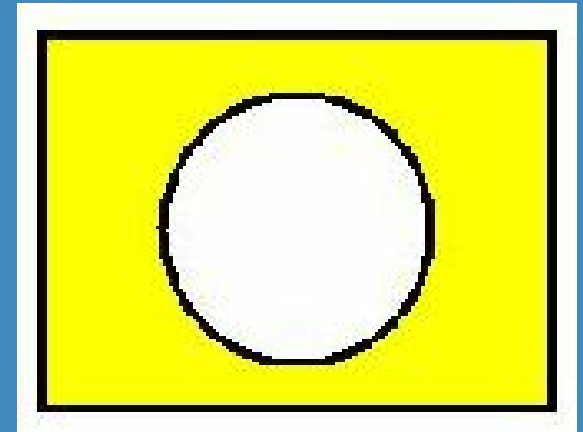
# Venn-diagrammal



AND



OR



NOT

# Példa

- If (i > 2) And (i < 8) Then

...

End If

Tipikus hiba: If i > 2 And < 8 Then

...

End If



# Még a logikai operátorokról

De Morgan-szabály:

Not (A And B)



(Not A) Or (Not B)

Not (A Or B)



(Not A) And (Not B)

# Feltételes utasítás végrehajtás

- Megkérdezzük a gyerektől az év végi bizonyítványának az átlagát. Ha 4.5, vagy annál jobb átlaga van, veszünk neki egy biciklit, egyébként nem veszünk.

Option Explicit

```
Private Sub cmdBicikli_Click()  
    Dim Atlag As Double  
    Atlag = InputBox("No fiam, milyen lett a bizonyítványod?")  
    If (Atlag >= 4.5) Then  
        MsgBox ("Gyere, veszünk neked egy bicajt!")  
    Else  
        MsgBox ("Hmmm, úgy látom, idén sem kapsz bicajt!")  
    End If  
End Sub
```

```
Private Sub cmdKilepes_Click()  
    Unload Me  
End Sub
```



# Beolvasás (InputBox)

Használata:

**változó = InputBox(„Üzenet”)**

Végrehajtásakor megjelenik az Üzenet, majd megjelenik egy beviteli sor, ahova be lehet gépelni, majd az OK megnyomásakor a beírt érték átadódik az egyenlőségjel bal oldalán álló változóba.

# InputBox

IF utasítás

Lesz-e biciklid?

Kilépés

Microsoft Excel

No fiam, milyen lett a bizonyítványod?

OK

Cancel

Microsoft Excel

No fiam, milyen lett a bizonyítványod?

OK

Cancel

4.7

Microsoft Excel

No fiam, milyen lett a bizonyítványod?

OK

Cancel

3.7

Microsoft Excel

Gyere, veszünk neked egy bicajt!

OK

Microsoft Excel

Hmmm, úgy látom, idén sem kapsz bicajt!

OK

# Gyakori feladat, több eset vizsgálata

- Beolvasni egy hallgató eredményét, majd a jegytől függően szövegesen kiírni:

1 – elégtelen

2 – elégséges

3 – közepes

4 – jó

5 – jeles

# Egy lehetséges megoldás, If-ekkel

```
Private Sub cmdJegy_Click()  
    Dim Jegy As Byte  
    Jegy = InputBox("Kérem az Ábrázoló ZH eredményét!")  
    If (Jegy = 5) Then  
        MsgBox ("Jeles")  
    End If  
    If (Jegy = 4) Then  
        MsgBox ("Jó")  
    End If  
    If (Jegy = 3) Then  
        MsgBox ("Közepes")  
    End If  
    If (Jegy = 2) Then  
        MsgBox ("Elégséges")  
    End If  
    If (Jegy = 1) Then  
        MsgBox ("Elégtelen")  
    End If  
End Sub
```

# Egy másik lehetséges megoldás, ElseIf-ekkel

```
Private Sub cmdJegy_Click()  
    Dim Jegy As Byte  
    Jegy = InputBox("Kérem az Ábrázoló ZH eredményét!")  
    If (Jegy = 5) Then  
        MsgBox ("Jeles")  
    ElseIf (Jegy = 4) Then  
        MsgBox ("Jó")  
    ElseIf (Jegy = 3) Then  
        MsgBox ("Közepes")  
    ElseIf (Jegy = 2) Then  
        MsgBox ("Elégséges")  
    ElseIf (Jegy = 1) Then  
        MsgBox ("Elégtelen")  
    End If  
End Sub
```

# Szétválogatásra: Select Case

- Kifejezetten az előző típusú feladatokra fejlesztették ki a Select utasítást. Használata:

**Select Case** *vizsgált kifejezés*

**Case** *érték*

*Utasítás1*

**Case** *érték*

*Utasítás2*

**Case Else**

*Utasítás3*

**End Select**

# Lehetséges megoldás

## Select Case-zel

```
Private Sub cmdJegy_Click()  
    Dim jegy As Byte  
    jegy = InputBox("Hogy sikerült a felelet?")  
    Select Case jegy  
        Case 5  
            MsgBox ("Jeles")  
        Case 4  
            MsgBox ("Jó")  
        Case 3  
            MsgBox ("Közepes")  
        Case 2  
            MsgBox ("Elégséges")  
        Case 1  
            MsgBox ("Elégtelen")  
    End Select  
End Sub
```

# Select Case variációk

```
Private Sub cmdJegy_Click()  
    Dim Jegy As Byte  
    Jegy = InputBox("Kérem az Ábrázoló ZH eredményét!")  
    Select Case Jegy  
        Case Is < 2  
            MsgBox ("Elégtelen")  
        Case 2  
            MsgBox ("Elégésges")  
        Case 3 To 4  
            MsgBox ("Alakul")  
        Case 5  
            MsgBox ("Jeles")  
        Case 6, 7, 8  
            MsgBox ("Nem kicsi család...")  
        Case Else  
            MsgBox ("Erről meg jobb nem is beszélni")  
    End Select  
End Sub
```



# Tömbök

- Két gyereknél, öt terméknél még belefér hogy egyesével felsorolásra kerülnek, de 10-20-100 esetében már nem.

```
Private Sub cmdTomb_Click()  
    Dim OsszSuly As Integer  
    Dim Suly1 As Integer, Suly2 As Integer, Suly3 As Integer  
    Dim Suly4 As Integer, Suly5 As Integer, Suly6 As Integer  
  
    OsszSuly = Suly1 + Suly2 + Suly3 + Suly4 + Suly5 + Suly6  
  
    MsgBox ("Összsúly:" & OsszSuly)  
End Sub
```

- Erre fejlesztették ki a tömböket

# Tömbök

- **Dim Suly(100) As Integer**
  - Ennek a tömbnek 101 eleme lesz
  - A tömb elemeire a számával lehet hivatkozni
  - Az első elem indexe nem 1, hanem 0, és minden elem indexe a sorszámánál eggyel kisebb lesz. Ez hibázási lehetőség, ezért:
- **Option Base 1** utasítással át lehet állítani 1-re.
  - Ez ún. modul szintű utasítás. Emiatt az eddig írt programokon kívül kell legyen.

# Dim utasítás tömbök esetében

Dim Ar (100) As Double (100 elem? 101 elem?)

Dim Ar (1 To 100) As Double (100 elem)

---

Dim Homerseklet (1 To 31, 1 To 24) As Double

Dim Homerseklet (31, 24) As Double

---

Dim RubikKocka (4, 4, 4) As Byte



# Tömbök

- A 27. személy súlyának megadása:

$$\text{Suly}(27) = 83$$

Most még nem látszik az előnye, hisz:

$$\text{OsszSuly} = \text{Suly}(1) + \text{Suly}(2) + \text{Suly}(3) + \text{Suly}(4) \dots$$

Sőt, látszólag még többet kell írni (és tényleg...)

De erre találták ki a ciklusokat.

# Tömbök

- Egy tömb tetszőleges elemére annak számával (Option Base 1 esetén sorszámmal) hivatkozhatok.
- Ezt a számot **index**-nek nevezzük.
- Az indexet azonban el lehet tárolni egy változóban.
- Pl.:  
 $i = 27$   
 $Suly(i) = 83$
- Ha a zárójelen belüli számot valamilyen művelet segítségével állítjuk elő, akkor azt kifejezésnek, **index kifejezésnek** hívjuk.

# Tömbök

Pl.:  $i = 27$

$Suly(i + 1) = 66$

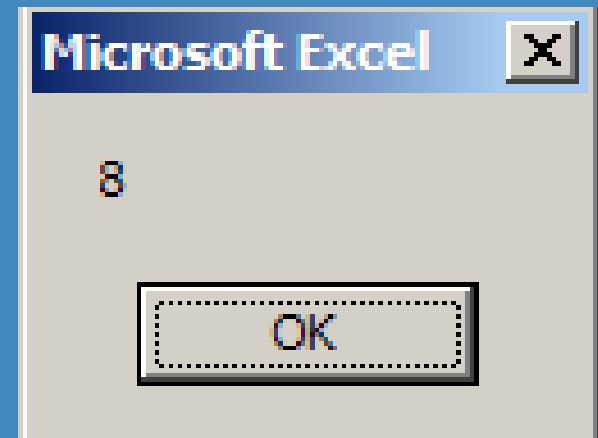
Ha a változónak megváltoztatjuk az értékét, a **korábbi értéke elveszik!**

Mobil telefon esetén is, ha egy ismerős telefonszámát felülírjuk, a korábbi szám elveszik.

$i = 27$

$i = 8$

MsgBox (i) →



# Ciklusok

Akkor használunk ciklust, ha többször, egymás után, ismétlődő tevékenységet kell végrehajtani a programban.

Lehet:

- Taxatív
- Iteratív

A feltétel vizsgálata szerint van:

- Elől tesztelő
- Hátul tesztelő

Nézzük táblánál...

# Ciklusok (2)

## Részei:

előkészítő rész

ciklus mag

végértékkel történő összehasonlítás

döntés

módosítás

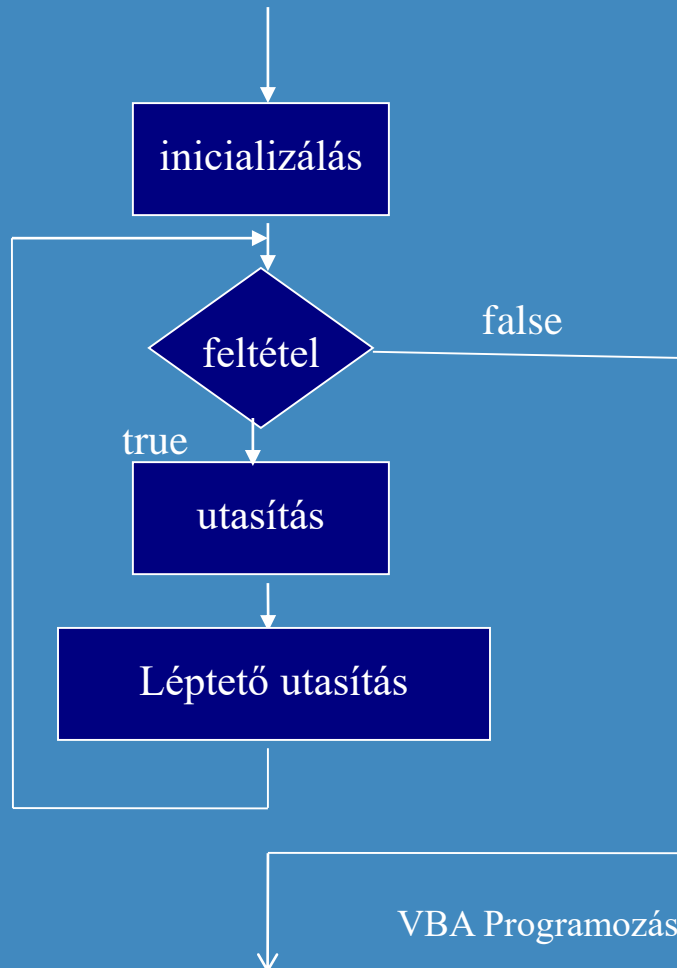
helyreállítás

Ciklusszámláló esetében **nem elfelejteni** a kezdőértékadást!



# For utasítás

- Alkalmazási területe: elsősorban adatsorozatok feldolgozása esetén
- Folyamatábrával:



# For utasítás

**For** *változó* = *Kezdeti\_érték* **To** *Vég\_érték* [**Step** *Lépésköz*]

*Utasítás*

**Next** [*változó*]

*A lépésköz megadása elmaradhat.*

*Ebben az esetben az értéke: +1*

*Működése:* az előző slide-on látottaknak megfelelően.

Pozitív lépésköz esetén: **kezdeti érték**  $\leq$  **végérték**

Negatív lépésköz esetén: **kezdeti érték**  $\geq$  **végérték**

# For utasítás (1)

```
For i = 1 To 10  
    Cells(i, 1) = i  
Next i
```

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

# For utasítás (2)

```
For i = 1 to 5
```

```
Cells(i, 1) = i
```

```
i = i + 1
```

```
Next i
```

	A	
1	1	
2		
3	3	
4		
5	5	
6		

Helyesen:

```
For i = 1 to 5 Step 2
```

```
Cells(i, 1) = i
```

```
Next i
```


!! (For) ciklusban a ciklusváltozó módosítása **szigorúan tilos!**

!! (For) ciklusból kiugrani **szigorúan tilos!** (talán később...)

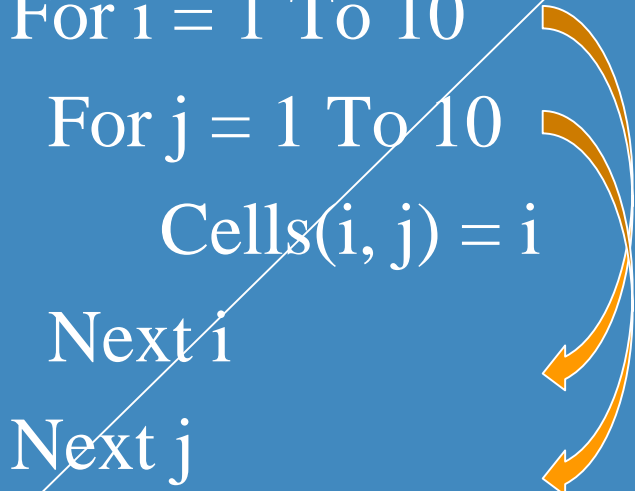
!! (For) ciklusba beugrani **szigorúan tilos!**

# For utasítás (3)

```
For i = 1 To 10  
  For j = 1 To 10  
    Cells (i, j) = i  
  Next j  
Next i
```



```
For i = 1 To 10  
  For j = 1 To 10  
    Cells(i, j) = i  
  Next i  
Next j
```



# For utasítás (4)

```
Private Sub cmdFor_Click()  
    Dim i As Integer  
    For i = 5 To 2  
        MsgBox (i)  
    Next i  
End Sub
```



```
Private Sub cmdFor_Click()  
    Dim i As Integer  
    For i = 5 To 2  
        MsgBox (i)  
    Next i  
End Sub
```



```
Private Sub cmdFor_Click()  
    Dim i As Integer  
    For i = 5 To 2  
        MsgBox (i)  
    Next i  
End Sub
```



```
Private Sub cmdFor_Click()  
    Dim i As Integer  
    For i = 5 To 2  
        MsgBox (i)  
    Next i  
End Sub
```

Lépésenként futtatva a programot, lehet látni, hogy a ciklusmagot **egyszer sem!** hajtja végre.

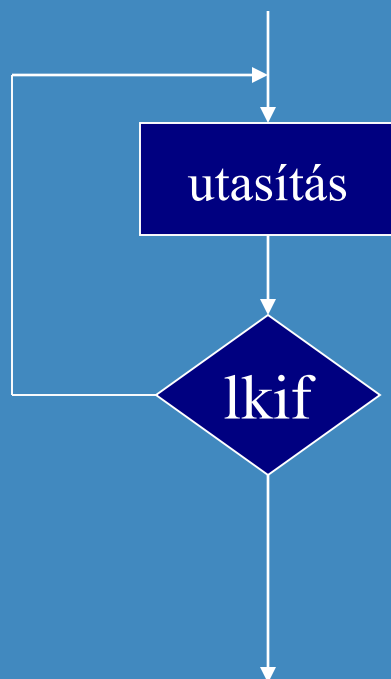
# Iteratív ciklus

Adjuk össze az egymást követő számokat 1-től kezdődően mindaddig, amíg az összeg el nem éri, vagy meg nem haladja a 10 000-et. Hány számot kell összeadni?

Erre való a Do ... Loop utasítás

# Folyamatábra jelekkel

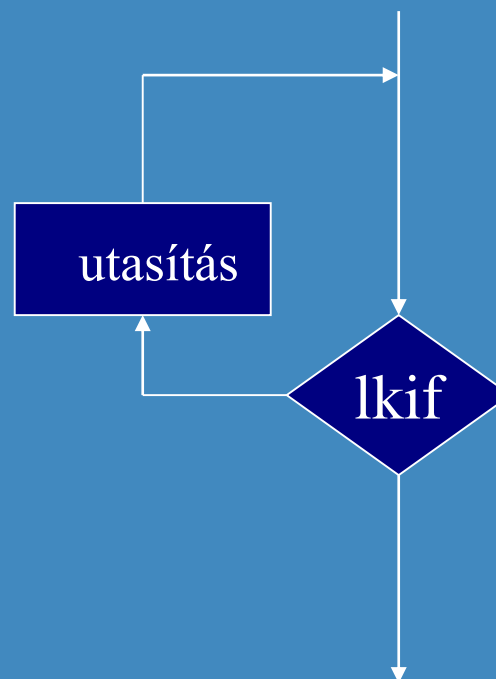
háttesztelő



Do ... Loop Until (logkif)

Do ... Loop While (logkif)

előtesztelő



Do Until (logkif) ... Loop

Do While (logkif) ... Loop



# Do ... Loop Until utasítás

**Do**

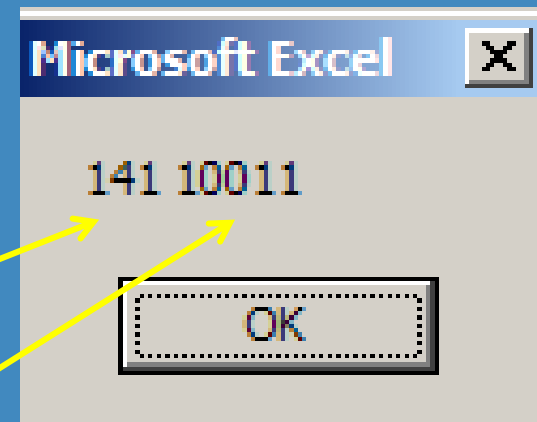
*Utasítás*

**Loop Until** (*logkif*)

Működése: végrehajtja a Do és a Loop közti utasítás(oka)t, majd megvizsgálja, hogy az Until mögötti logikai kifejezés igaz-e. Ha **hamis**, akkor újra megismétli az utasítás(oka)t. Mindaddig, amíg az hamis. Ezért az **kilépési feltétel**.

# Do ... Loop Until utasítás

```
Private Sub cmdCiklus_Click()  
    Dim Osszeg As Long  
    Dim i As Integer  
    Osszeg = 0  
    i = 0  
    Do  
        i = i + 1  
        Osszeg = Osszeg + i  
    Loop Until Osszeg >= 10000  
    MsgBox (i & " " & Osszeg)  
End Sub
```



# Do ... Loop While utasítás

**Do**

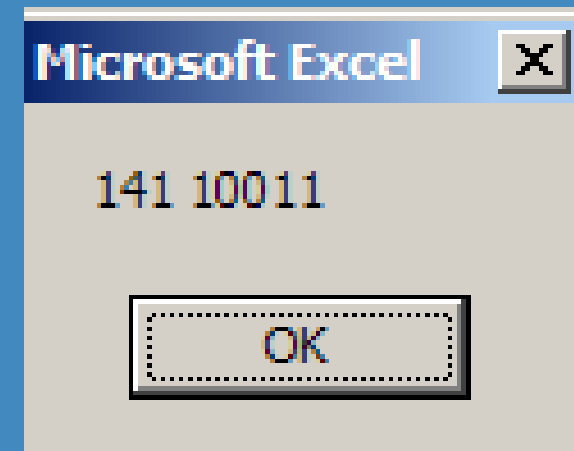
*Utasítás*

**Loop While** (*logkif*)

Működése: végrehajtja a Do és a Loop közti utasítás(oka)t, majd megvizsgálja, hogy a While mögötti logikai kifejezés igaz-e. Ha **igaz**, akkor újra megismétli az utasítás(oka)t. Mindaddig, amíg az igaz. Ez is **kilépési feltétel**.

# Do ... Loop While utasítás

```
Private Sub cmdCiklus_Click()  
    Dim Osszeg As Long  
    Dim i As Integer  
    Osszeg = 0  
    i = 0  
    Do  
        i = i + 1  
        Osszeg = Osszeg + i  
    Loop While Osszeg < 10000  
    MsgBox (i & " " & Osszeg)  
End Sub
```



# Do ... Loop Until

# Do ... Loop While

- Mindkettő hátul tesztelő (az utasítás után ellenőriz)
- Az egyikből a másik gyorsan előállítható, csak a relációjelet kell megfordítani.
- < helyett >=
- > helyett <=
- <= helyett >
- >= helyett <
- = helyett <>

# Do Until ... Loop

## Do Until ... While

- Mindkettőnél a feltétel ellenőrzése az utasítás előtt van, ezért mindkettő elől tesztelő.

# Do Until ... Loop

**Do Until** (*logkif*)

*Utasítás*

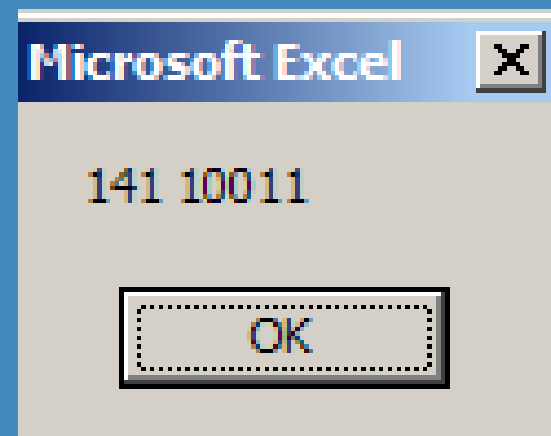
**Loop**

Működése: megvizsgálja, hogy az Until mögötti logikai kifejezés **hamis-e**, Ha igen, végrehajtja az Utasítást, majd újra vizsgál. Mindaddig, amíg hamis, ismétel. Ha igaz lesz, kilép..

Lehet, hogy egyszer sem hajtja végre! **Belépési feltétel.**

# Do Until ... Loop

```
Private Sub cmdCiklus_Click()  
    Dim Osszeg As Long  
    Dim i As Integer  
    Osszeg = 0  
    i = 0  
    Do Until Osszeg >= 10000  
        i = i + 1  
        Osszeg = Osszeg + i  
    Loop  
    MsgBox (i & " " & Osszeg)  
End Sub
```





# Do While ... Loop

**Do While** (*logkif*)

*Utasítás*

**Loop**

Működése: megvizsgálja, hogy a While mögötti logikai kifejezés **igaz-e**. Mindaddig, amíg igaz, ismétel. Ha hamis lesz, abbahagyja az ismétlést.

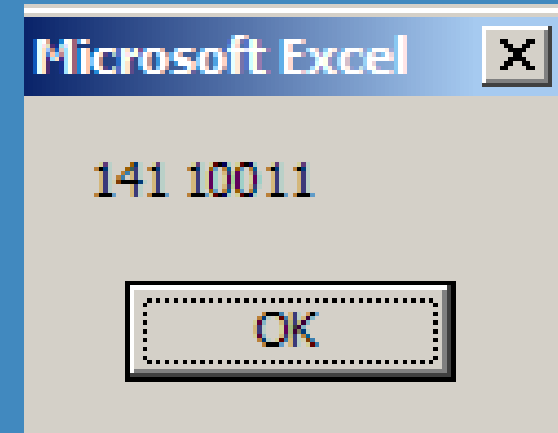
Ha az elején már hamis, egyszer sem hajtja végre.

**Belépési feltétel.**

VBA Programozás

# Do While ... Loop

```
Private Sub cmdCiklus_Click()  
    Dim Osszeg As Long  
    Dim i As Integer  
    Osszeg = 0  
    i = 0  
    Do While Osszeg < 10000  
        i = i + 1  
        Osszeg = Osszeg + i  
    Loop  
    MsgBox (i & " " & Osszeg)  
End Sub
```



# Felhasználó által definiált típus

Egy személy adatait szeretnénk tárolni.

Név	String
Fizetés	Long
Gyerekek száma	Byte
...	

Ilyen kevert típus azonban nincs, létre kell hozni!

(Ráadásul ha szükség van a vezetéknévére, akkor érdemes a nevet kettébontva tárolni)

# Felhasználó által definiált típus

Használata:

Private Type Ember

Vezeteknev As String

Keresztnev As String

Fizetes As Long

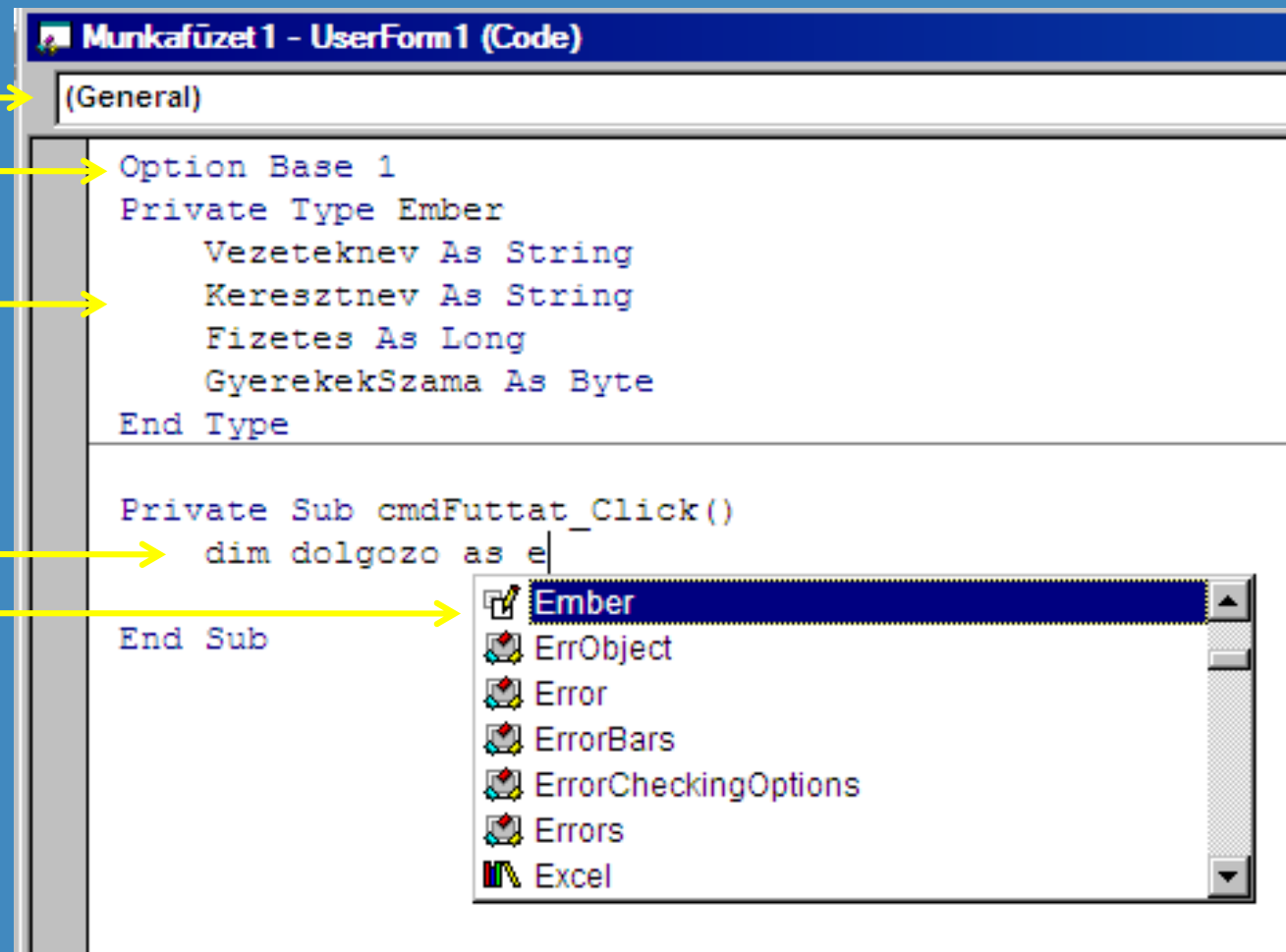
GyerekekSzama As Byte

End Type

Ez még CSAK típus!!

# Felhasználó által definiált típus

Az Option Base 1-hez hasonlóan modul szintű utasítás, vagyis a programon kívül kell megadni



# Felhasználó által definiált típus

Dim Dolgozo As Ember

Értékkadás:

Változó neve

Mező neve

Mező értéke

Dolgozo.VezetekNev = "Kovács"

Dolgozo.Keresztnev = "János"

Dolgozo.Fizetes = 320000

Dolgozo.GyerekekSzama = 2

# Tömb + Felhasználó által definiált típus

Dim Dolgozok(100) As Ember

Dolgozok(5).VezetekNev = "Kempelen"

Dolgozok(5).Keresztnev = "Farkas"

Dolgozok(5).Fizetes = 360000

Dolgozok(5).GyerekekSzama = 0

# Alapalgoritmusok

- Számlálás
- Összegzés
- Átlagolás
- Osztályba sorolás
- Minimum-maximum keresés
- Rendezés

Az alapalgoritmusok szerepe...



# Minta feladat

Meg kell határozni egy vállalat dolgozóinak átlagfizetését, majd kikeresni a legkisebb fizetésű dolgozót, és megemelni a fizetését az átlagra.

1. Az átlagfizetéshez először **összegezni** kell a dolgozók fizetését
2. Majd meg kell számolni hány dolgozó van (**számlálás**)
3. El kell osztani az összeget a dolgozók létszámával (**átlag**)
4. Megkeresni a legkisebb fizetésű dolgozót (**minimumkeresés**)
5. Megemelni a fizetését

Egy program tehát (többnyire) építőelemekből áll.

# Számlálás

Előírt feltételnek eleget tevő esetek megszámlálása.

A számlálás gyűjtő algoritmus, mivel az előfordult esetek darabszámát egy gyűjtő változóban tartja.

Minden gyűjtő algoritmus fontos eleme, hogy a gyűjtés (összegzés) előtt a gyűjtő változót le kell nullázni.

Tipikus hiba: nulláznak, de a ciklusban!

VBA Programozás

# Számlálás

```
cmdKilepes  Click

Option Explicit

Private Sub cmdKilepes_Click()
    Unload Me
End Sub

Private Sub cmdSzamlalas_Click()
    'Az Excel tábla első oszlopában levő 10 érték közül hány pozitív van?
    Dim i As Integer
    Dim Db As Integer
    Db = 0
    For i = 1 To 10
        If Cells(i, 1) > 0 Then
            Db = Db + 1
        End If
    Next i
    MsgBox ("A pozitív elemek darabszáma: " & Db)
End Sub
```

# Számlálás

The image shows a Microsoft Excel spreadsheet with a UserForm and a message box overlaid on it.

**Excel Spreadsheet:**

A	B	G
1		
2		
-3		
4		
5		
-6		
-7		
8		
7		
6		

**UserForm1:**

- Buttons: Számlálás, Összegzés, Átlagolás, Kilépés

**Microsoft Excel Message Box:**

A pozitív elemek darabszáma: 7

OK

# Összegzés

Feltétel nélkül, vagy feltételtől függően numerikus értékek összegét képezzük.

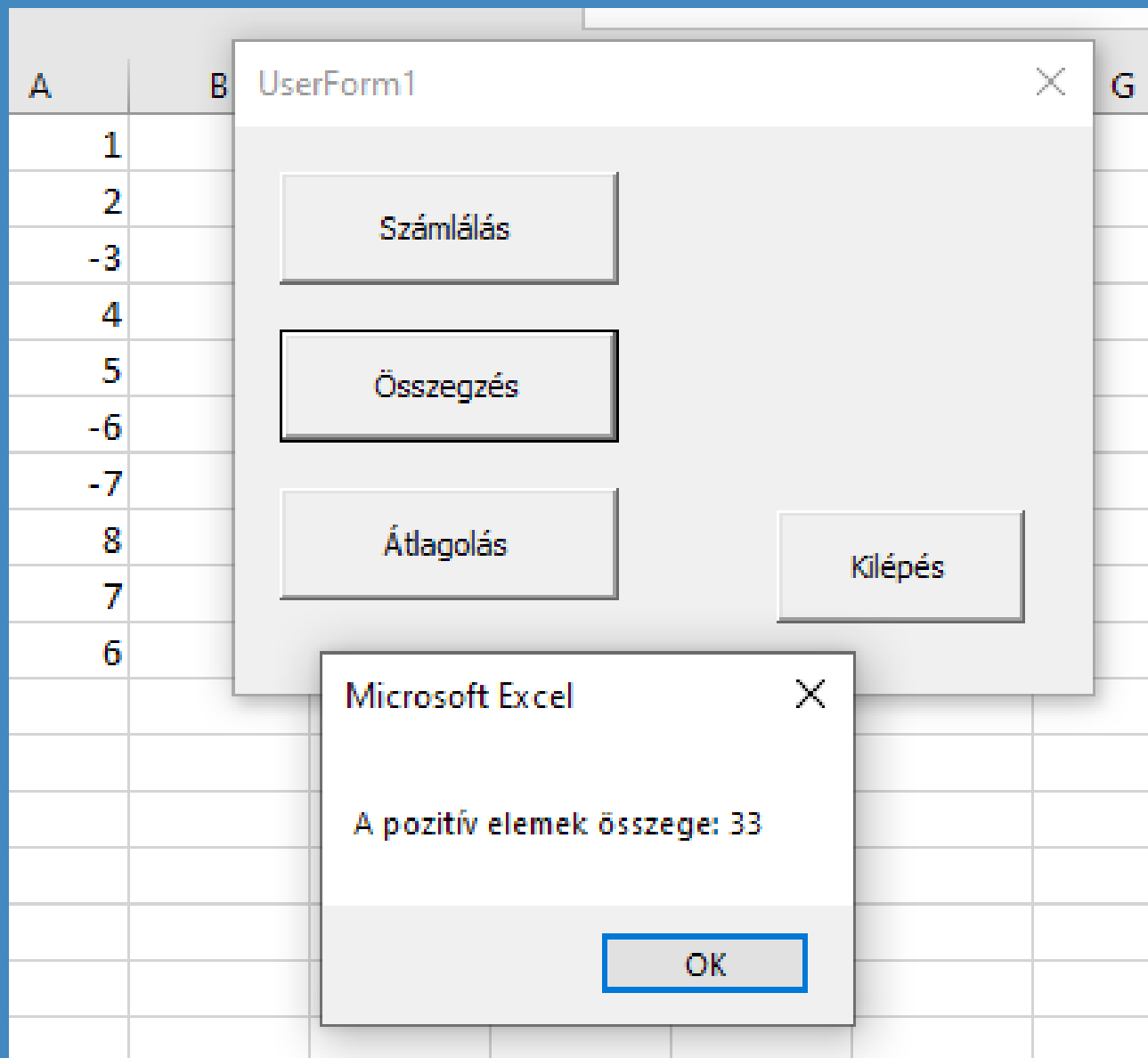
Gyűjtő algoritmus, gyűjtő változó, nullázás.

Különbség a számláláshoz képest, hogy a gyűjtő változó értékét nem 1-gyel növeljük, hanem magával az értékkel.

# Összegzés

```
Private Sub cmdOsszegzes_Click()  
'Az Excel tábla első oszlopában levő 10 érték közül a pozitívak összege?  
    Dim i As Integer  
    Dim Osszeg As Integer  
    Osszeg = 0  
    For i = 1 To 10  
        If Cells(i, 1) > 0 Then  
            Osszeg = Osszeg + Cells(i, 1)  
        End If  
    Next i  
    MsgBox ("A pozitív elemek összege: " & Osszeg)  
End Sub
```

# Összegzés



# Átlagolás

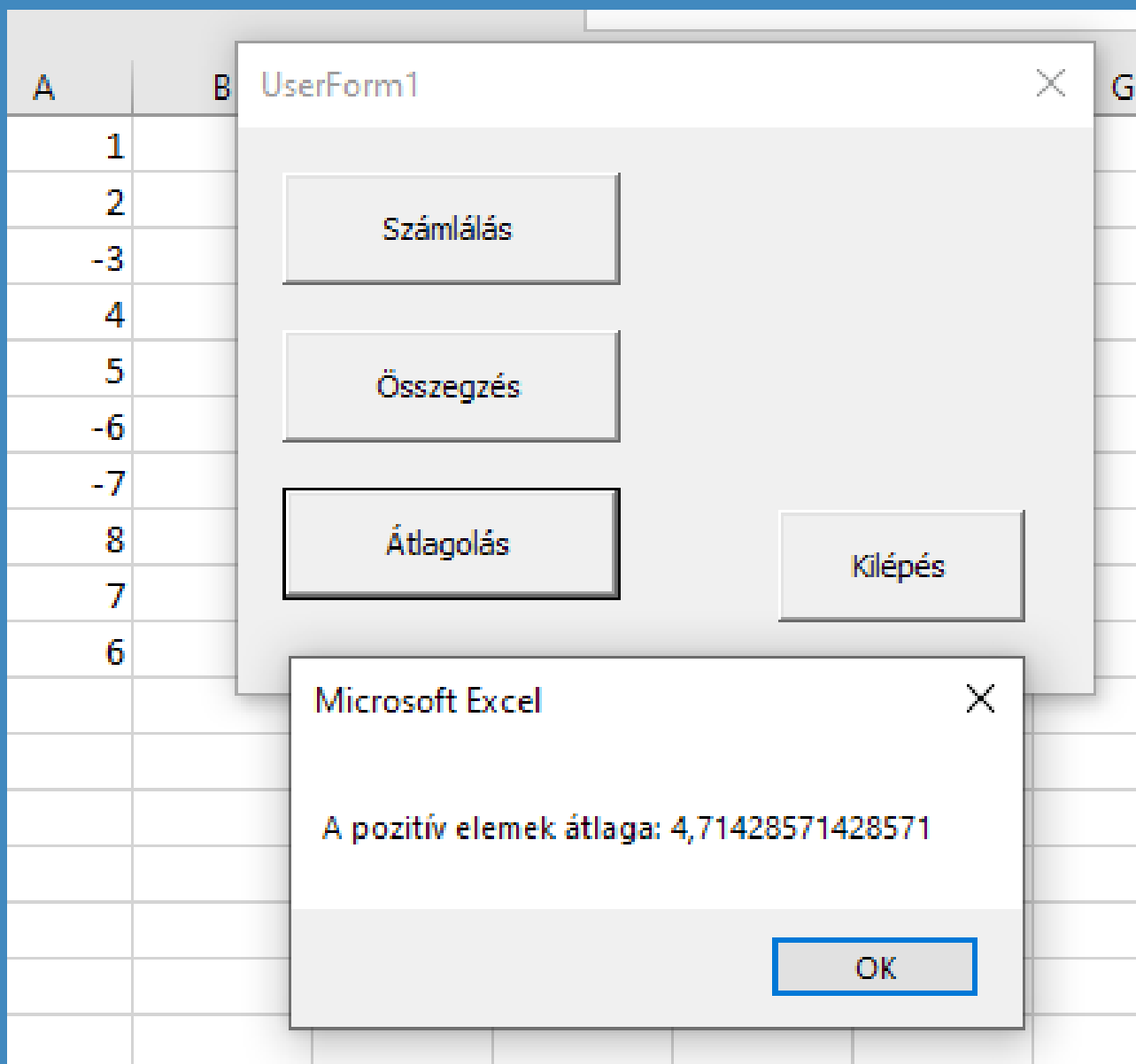
- Az összeget el kell osztani a darabszámmal.
- Hiba lehetőség: ha nem volt megfelelő eset, akkor nullával osztunk!! Osztás előtt vizsgálni!
- Ügyelni a típusra, Integer típus a legritkább esetben jó csak. Inkább valós (Double) legyen.



# Átlagolás

```
Private Sub cmdAtlagolas_Click()  
'Az Excel tábla első oszlopában levő 10 érték közül a pozitívak átlaga?  
Dim i As Integer  
Dim Db As Integer, Osszeg As Integer, Atlag As Double  
Db = 0  
Osszeg = 0  
For i = 1 To 10  
    If Cells(i, 1) > 0 Then  
        Db = Db + 1  
        Osszeg = Osszeg + Cells(i, 1)  
    End If  
Next i  
If Db > 0 Then  
    Atlag = Osszeg / Db  
    MsgBox ("A pozitív elemek átlaga: " & Atlag)  
Else  
    MsgBox ("Nem volt pozitív elem, nem számolható átlag")  
End If
```

# Átlagolás



# Osztálybasorolás

Tipikus példa, év végi átlagok:

AH	FH	Osztály
2.00	2.50	elégséges
2.51	3.50	közepes
3.51	4.50	jó
4.51	5.00	jeles

Megvalósítása egyszerű esetben IF-ekkel, de ügyesebb, ha tömbben tároljuk az osztályok határait...

```

Private Sub cmdOsztalybaSorolas_Click()
    Dim OsztalyDb(5)
    Dim H(5) As Double
    Dim i As Integer, j As Integer
    H(1) = 2
    H(2) = 2.5
    H(3) = 3.5
    H(4) = 4.5
    H(5) = 5
    For i = 1 To 5
        OsztalyDb(i) = 0
    Next i
    For i = 1 To 10
        If (Cells(i, 1) >= H(1)) And (Cells(i, 1) < H(2)) Then
            '2 ... 2.5
            OsztalyDb(1) = OsztalyDb(1) + 1
        ElseIf (Cells(i, 1) >= H(2)) And (Cells(i, 1) < H(3)) Then
            '2.5 ... 3.5
            OsztalyDb(2) = OsztalyDb(2) + 1
        ElseIf (Cells(i, 1) >= H(3)) And (Cells(i, 1) < H(4)) Then
            '3.5 ... 4.5
            OsztalyDb(3) = OsztalyDb(3) + 1
        ElseIf (Cells(i, 1) >= H(4)) And (Cells(i, 1) < H(5)) Then
            '4.5 ... 5
            OsztalyDb(4) = OsztalyDb(4) + 1
        ElseIf (Cells(i, 1) = H(5)) Then
            '5.00 kitűnő
            OsztalyDb(5) = OsztalyDb(5) + 1
        End If
    Next i
    For i = 1 To 5
        Cells(2 + i, 3) = OsztalyDb(i)
    Next i
End Sub

```

# Osztálybasorolás

A	B	C	D	E	F	G	H	I
2,13								
2,15								
3,76	Elégséges	2						
3,92	Közepes	1						
3,59	Jó	4						
4,44	Jeles	1						
2,64	Kiváló	1						
5,00								
4,87								
4.67								

Osztálybasorolás

Osztálybasorolás

Kilépés

# Minimum - maximum keresés

- Egy dobozba golyókat dobálnak.
  - Minden golyón van egy szám.
  - Egyszerre két golyót tudunk kivenni (két kéz van)
  - Melyik a legkisebb (v. legnagyobb) szám?
- 
- Az első ötletek: If-ekkel

# Minimum – maximum keresés

2 adat

```
If (a > b) Then  
    MsgBox a  
Else  
    MsgBox b  
End If
```

3 adat

```
If (a > b) And (a > c) Then  
    MsgBox (a)  
ElseIf (b > a) And (b > c) Then  
    MsgBox (b)  
ElseIf (c > a) And (c > b) Then  
    MsgBox (c)  
End If
```

4 – 5 – 100 érték esetén???

# Minimum – maximum keresés

- Teljesen véletlenszerűen vegyünk ki egy golyót a dobozból.
- Állítsuk róla, hogy az a legkisebb.
- Hasonlítsuk össze a következővel.
- Ha ez kisebb, akkor megjegyezzük a sorszámát, és innentől kezdve hozzá viszonyítunk.
- Ha nem kisebb, akkor továbbra is az elsőhöz.
- A legvégén tudjuk a legkisebb elem sorszámát.



# Minimum keresés

```
Private Sub cmdMinimum_Click()  
    Dim Mini As Integer, i As Integer  
    Mini = 1  
    For i = 2 To 10  
        If Cells(i, 1) < Cells(Mini, 1) Then  
            Mini = i  
        End If  
    Next i  
    MsgBox ("A legkisebb elem indexe: " & Mini & " értéke: " & Cells(Mini, 1))  
End Sub
```

# Minimum keresés

	A	B	C	D	E	F
1	43906,00					
2	4,15					
3	3,76					
4	3,92					
5	3,59					
6	4,44					
7	2,64					
8	5,00					
9	4,87					
10	4.67					
11						
12						
13						
14						
15						

Osztálybesorolás

Minimum keresés

Microsoft Excel

A legkisebb elem indexe: 7 értéke: 2,64

OK

# Minimum keresés

- Fontos!
- Soha se értéket jegyezzünk meg, mindig sorszámot!
- A sorszám ismeretében bármikor megkapjuk az értéket,
- Érték ismeretében még ki kellene keresni, melyik elemnek ennyi az értéke

# Rendezés

## Az alapgondolat:

- A rendezett listában a legelső elem a legkisebb.
- A második elem a második legkisebb, de az első nem nézve a legkisebb. Másképp fogalmazva az elsőől jobbra levők közül a legkisebb.
- ... és ez minden elemre igaz...
- A módszer tehát: minimum keresés.
- A megtalált legkisebbet a neki megfelelő helyen levővel felcseréljük.

# Rendezés

- Az első helyen álló elemről azt állítjuk, hogy a tőle jobbra levő elemek közül ő a legkisebb.
- Minimum keresés.
- A megtalált legkisebbet megcseréljük az első helyen levővel.
- Most az egészet megismételjük a második helyen levővel.
- Majd a harmadikon levővel, egész végig... (?)

Ciklus az 1.  
elemtől az  
utolsó előttiig

**For i = 1 To n-1**

Állítás

**mini = i**

Minimum  
keresés

**For j = i + 1 To n**  
    **If (t ( j ) < t ( mini )) Then**  
        **mini = j**  
    **End If**  
**Next j**

Csere

**s = t ( i )**  
**t ( i ) = t ( mini )**  
**t ( mini ) = s**

**Next i** VBA Programozás

# Rendezés programja

```
Dim Mini As Integer
Dim i, j As Integer
Dim Csere As Double
For i = 1 To 30 - 1
    Mini = i
    For j = i + 1 To 30
        If Cells(j, 1) < Cells(Mini, 1) Then
            Mini = j
        End If
    Next
    Csere = Cells(Mini, 1)
    Cells(Mini, 1) = Cells(i, 1)
    Cells(i, 1) = Csere
Next
txtSzamlalas = "Kész a rendezés!"
```

# Szubrutinok (1)

- **Jelenség**: ha ugyanazt a műveletsort kell elvégezni más és más adathalmazon, akkor jelenleg **többször ismétlődő programrészletet** kell írunk a programba.

## **Következmény**:

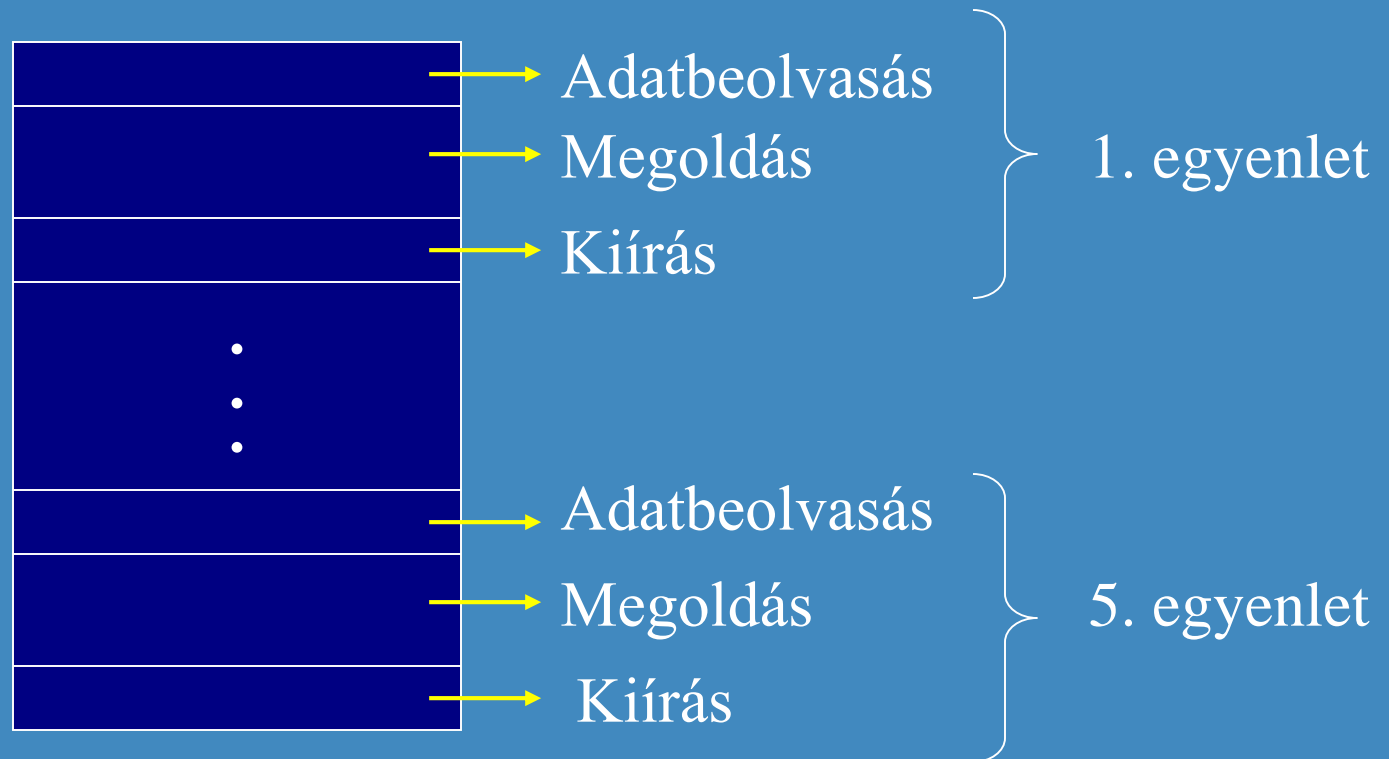
- sokat kell írni *(majd másolom az ismétlődő utasításokat)*
- hosszú lesz a program *(legalább látják mekkora munkám van)*
- sok helyet foglal a file a HDD-n *(na bumm, elfér, nem?)*
- tovább tart a program lefordítása *(kibírom.)*
- véletlenül hibásan írtam. **Hány helyen is kell javítanom?**



# Szubrutinok (2)

Például: 5 db másodfokú egyenletet kell megoldani. Variációk:

## 1. Nincs szubrutin

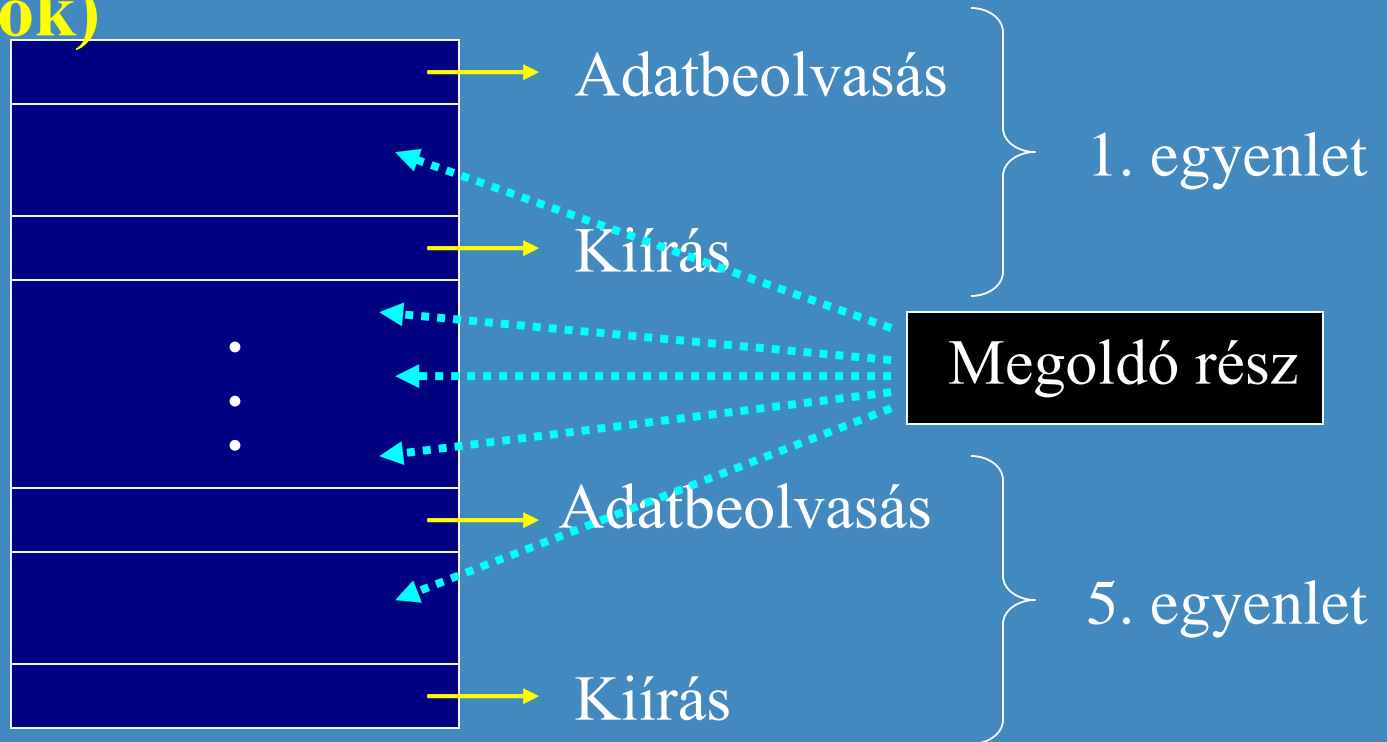


VBA Programozás

# Szubrutinok (3)

## 2. Nyílt szubrutin (makrók)

Szimbólummal jelölik, ez alapján hívják. A forrás programban csak egyszer van, de a lefordított programban annyiszor fordul elő, ahányszor meghívták.

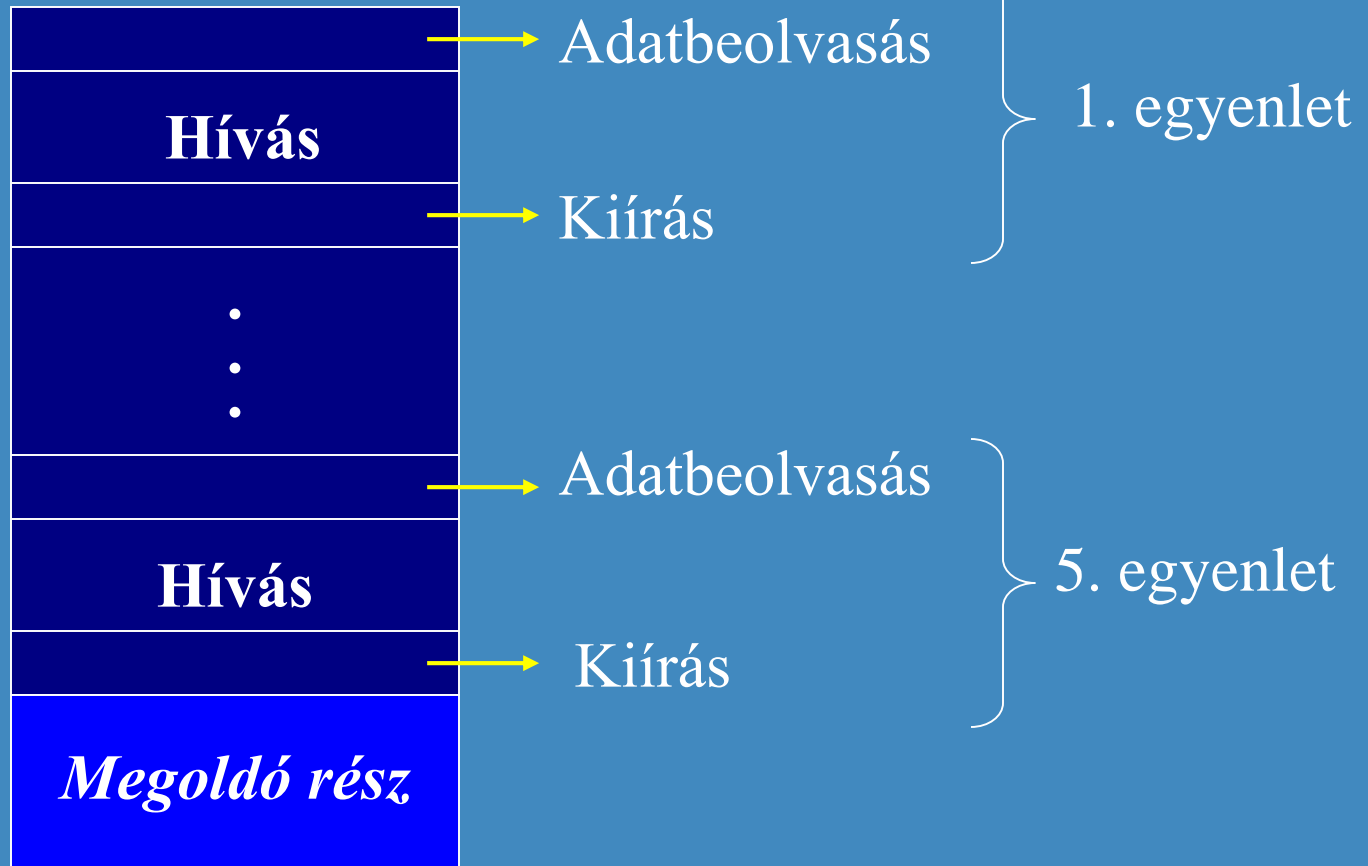


# Szubrutinok (4)

## 3. Zárt szubrutin

A hívó rutin területén kívül tárolódik. A forrásban, és a lefordított programban is csak egyszer fordul elő.

Kezdetben a tár fix helyén volt, később relokálható lett.



# Szubrutinok (5)

A függvénynek mindig van visszatérési értéke.

# Szubrutinok (6)

- A függvények tetszőleges sorrendben követhetik egymást, az egyetlen kikötés: **első felhasználásuk előtt deklarálni kell őket**. (Ez egyedül csak olyan esetben probléma, amikor szubrutinok egymást hívják, de akkor is megoldható a FORWARD segítségével).
- A szubrutinok a program önálló részei, melyeknek a futás során át lehet adni a vezérlést. Ezt úgy nevezzük (zsargon): **meghívjuk a szubrutint**. Ekkor a szubrutin utasításai hajtódnak végre, majd a végrehajtás a hívó program **hívást követő első utasításán** folytatódik.

# Szubrutinok (7)

- Paraméterek:

A szubrutinok paraméterek segítségével kommunikálhatnak környezetükkel. Az eljárások eredményüket is e paraméterek segítségével adják vissza a hívónak. **Nem globális változókkal dolgozni!!!**

**típus** FvNév (típus paraméter\_név)

```
{  saját típusok;  
    változók;  
    utasítások;  
    return ...;  
}
```

# Formális és aktuális paraméterek

A függvények fejlécében felsorolt paramétereket FORMÁLIS paramétereknek nevezzük.

Azokat a paramétereket pedig, amelyekkel a függvényt meghívjuk, AKTUÁLIS paramétereknek nevezzük.

A formális és aktuális paraméterek darabszámának, és páronként típusának meg kell egyeznie!

# Paraméter átadási módok

Kétféle paraméterátadási mód van:

- címszerűti
- értékszerűti

A paraméter átadás módját a függvény fejlécében lehet meghatározni.



# Címszerinti paraméterátadás

A fv hívásakor a paraméternek a címe kerül átadásra. A szubrutin a cím alapján tudja módosítani az átadott paraméter értékét. Látszólag úgy tűnik, hogy a megváltoztatott érték a hívó programba való visszatérés után is megmarad.

A szubrutin fejlécében a paraméter neve előtt ott szerepel a \*

# Értékszerinti paraméterátadás

A fv hívásakor a szubrutin is helyet foglal a memóriában a változónak, ahova a paraméterként **megkapott értéket bemásolja**. Ha a függvény megváltoztatja ennek a paraméternek az értékét, akkor a hívó programba való visszatérés után a megváltoztatott érték nem kerül vissza, hanem **elveszik**.

Értékszerinti paraméterátadáskor a paraméter lehet konstans vagy kifejezés is.

Lefutása után a szubrutin az általa lefoglalt memória területeket felszabadítja.

# Összehasonlítás

- Címszerinti:

- Előny:

- gyorsabb paraméterátadás, gyorsabb futás
    - rövidebb a lefordított program

- Hátrány:

- ha a szubrutin megváltoztatja a paraméter értékét, akkor az visszakerül a hívóprogramba (nem hátrány, következmény!!)

- Értékszerinti

- Előny:

- konstansok is átadhatók, így egyszerűbb a meghívás

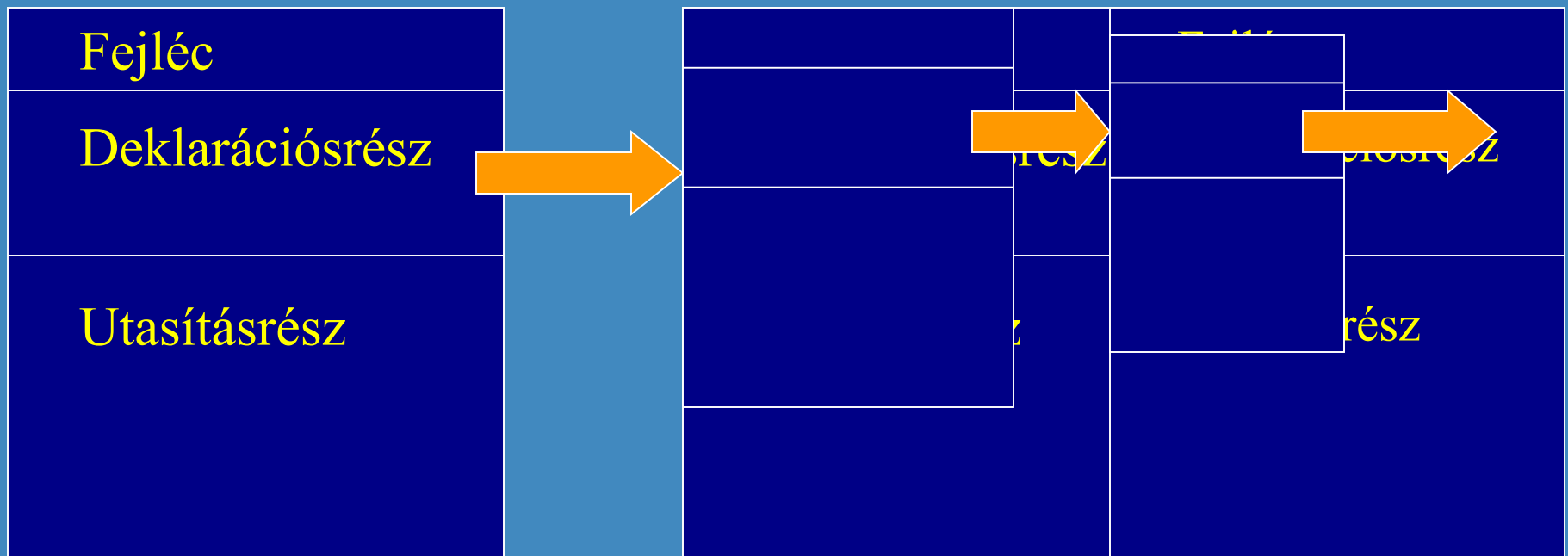
- Hátrány:

- Nagyobb futtatható programméret
    - lassúbb végrehajtás

# Azonosítók érvényességi köre

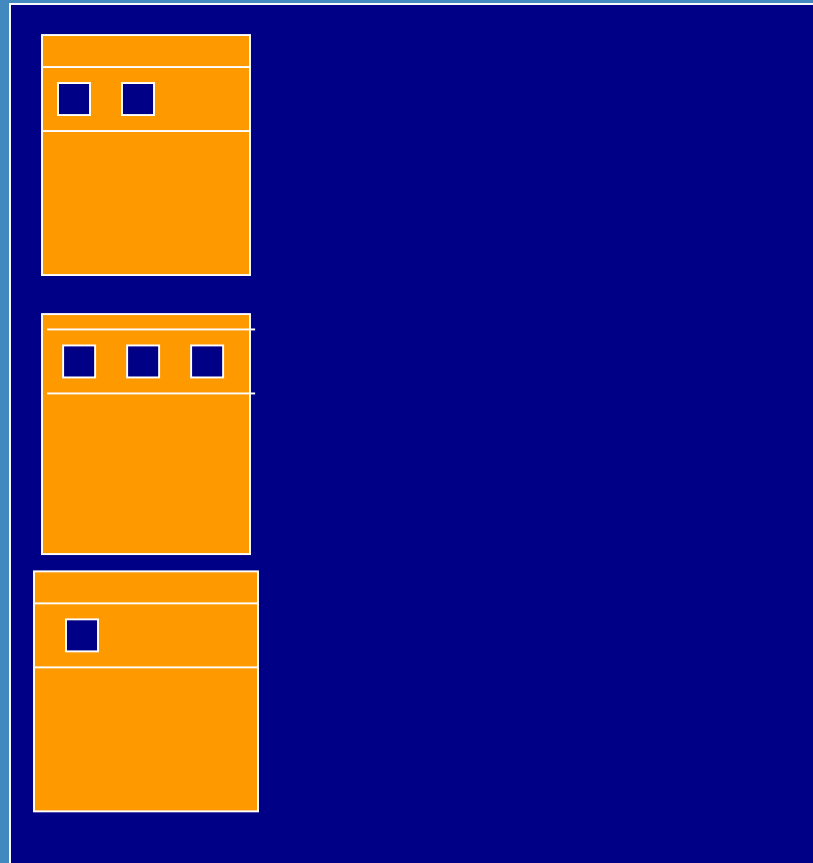
Kiindulási alap: a blokkszerkezetű programozási nyelvek (C, FORTRAN, PASCAL, ...)

Egy program szerkezete:



# Azonosítók érvényességi köre (2)

Minden egyes blokk tehát újabb blokk(ka)t tartalmazhat



VBA Programozás

# Azonosítók érvényességi köre (3)

A blokk deklarációs részében azonosítókat lehet deklarálni.

Az azonosítók és a blokkok viszonyától függően háromféle érvényességi kört különböztetünk meg.

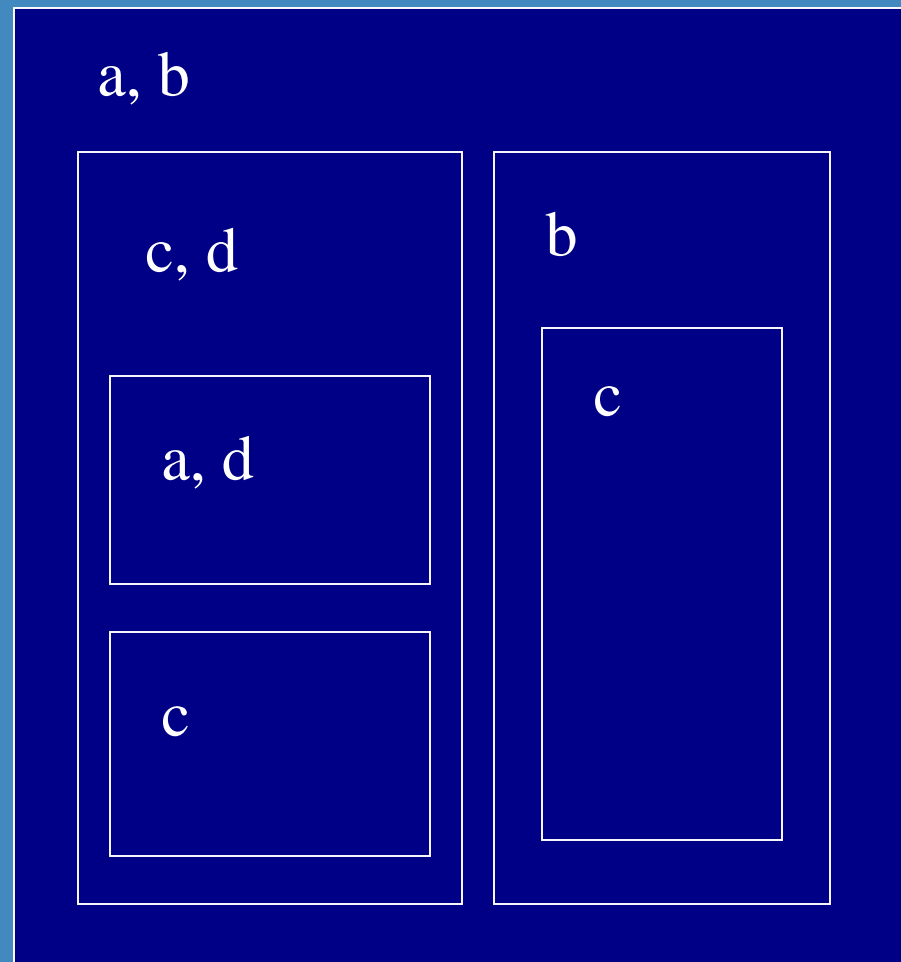
```
int j;
```

# Azonosítók érvényességi köre (4)

- Lokális: ...
- Globális: ...
- Láthatatlan: ...

**De!!** A globális azonosítókat NEM szeretjük!!

**Átláthatatlanná teszik a programot.**



VBA Programozás

# A Char típus (1)

- Értelmezési tartománya: az ASCII karakterkészlet. 0-255-ig.
- Sorszámozva (!) tartalmazza a karaktereket, azaz minden karakterhez hozzá van rendelve egy (szám)kód.
- A karakterek egy-egy byte-ot foglalnak el a memóriában. E byte értéke az adott karakter ASCII kódja, megjelenési formája (pl. képernyőn való kiíratás esetén) a karakter képe.



# A Char típus (2)

Használata:

...

```
char c;
```

```
c = 'x';
```

```
printf("c = %c\n", c);
```

```
scanf("%c", &c);
```

```
printf("c = %c\n", c);
```

...

Fontos: **egyetlenegy** karakter tárolására alkalmas!

# A sztringek (1)

- Vagy másképpen: változó hosszúságú karakter sorozat. (De nem dinamikus módon történik a helyfoglalás.)
- Hossza: 0 .. 255 karakter
- Összetett típus

- String konstans: “abc”
- String változó:

**char nev[30];**

**A C-ben (a Pascaltól eltérően) nincs önálló sztring típus, hanem karakter tömbként kell kezelni.**

**Egy fontos jellemzője van: mivel változó hosszúságú, a karaktersorozat végét valamilyen módon jelölni kell. C-ben: \0**

# A sztringek (2)

- Használatuk többnyire függvényekkel (`#include <string.h>`)
- Tárolása:

`strcpy(ss, “Ez egy példa szöveg”);`



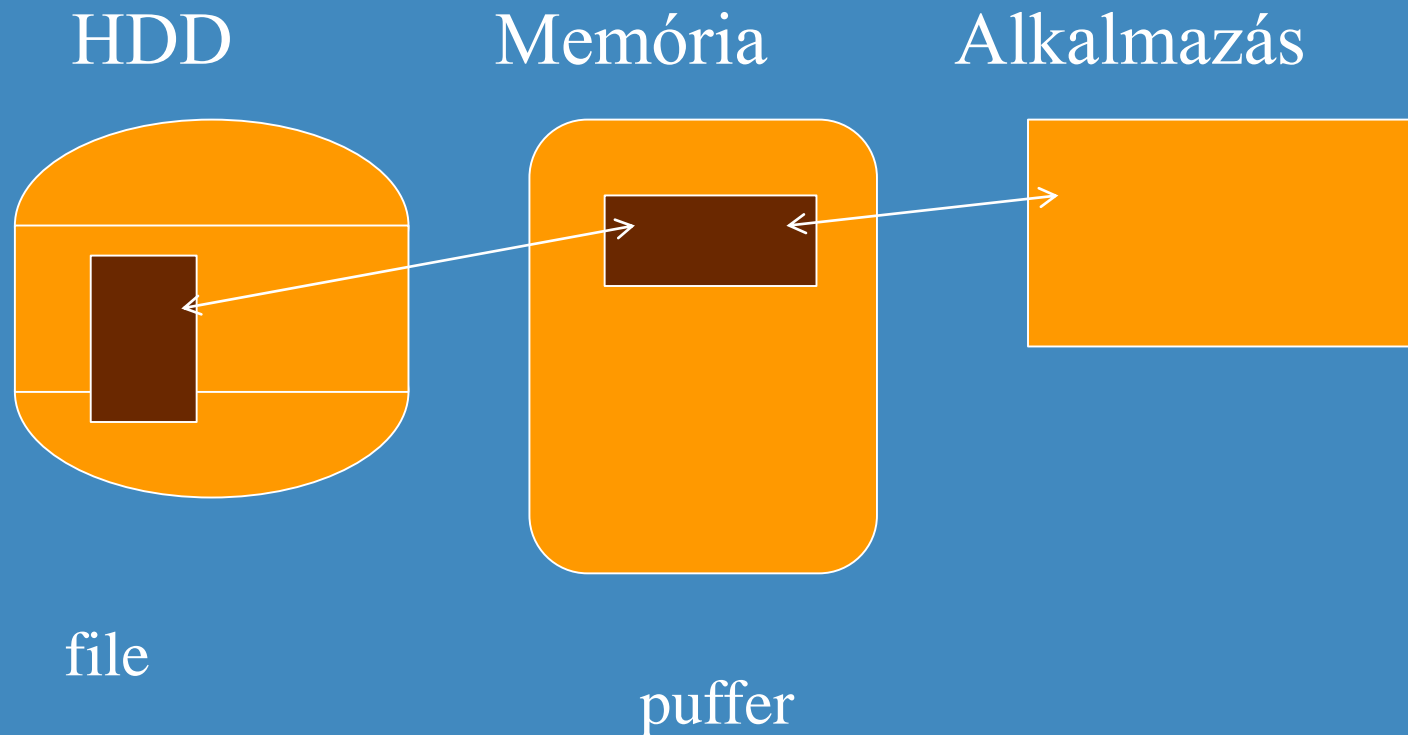
karakterkódok

# A sztringek (3)

- A tárolási mód miatt gondolni kell a véget jelző \0 karakterre is, emiatt ha a tárolni kívánt karaktersorozat 30 karakter hosszú lehet, a deklarálás ilyen kell legyen:
- `char nev[31];`

Összehasonlítás, értékadás, stb. függvényekkel történik.

# File kezelés C-ben (1)



# File kezelés C-ben (2)

A file kezeléséhez szükséges egy speciális változó:

- Típusa: **FILE \***
- Ez egy struktúrára mutató pointer
- Az stdio.h-ban definiálva
- Tagjai:
  - File sorszáma
  - FLAG byte (utolsó művelet utáni állapot. Pl: EOF, vagy error)
  - Puffer címe
  - Pufferben levő byte-ok száma
  - file pointer (a file-on belüli következő adathoz), ...

# File kezelés C-ben (3)

Valójában a FILE \* struktúra típusú tömbre mutat, Elemeinek száma 20.

Ebből következik, hogy egyidőben maximum 20 file lehet nyitva

Az első 5 foglalt:

stdin – standard input csatorna

stdout – standard output csatorna

stderr – standard error csatorna

stdaux – standard aszinkron átviteli csatorna

stdprn – standard nyomtató kimeneti csatorna

# File kezelés C-ben (4)

A file-t a használat előtt meg kell nyitni!

Az első szabad tömb elembe kerülnek bele a most megnyitott file adatai.

A file megnyitás paramétereai:

- filenév az útvonallal
- megnyitási mód



# File kezelés C-ben (5)

Filenév az útvonallal:

- Windows alatt: "meghajtó:\\útvonal\\filenév"
- Unix alatt: "\\útvonal\\filenév"

Megnyitási mód:

két adatot kell megadni:

- milyen jellegű a file (text, bináris)
- milyen módon kívánjuk használni (olvasás, írás)

# File kezelés C-ben (6)

- Szöveges file megnyitható text és bináris módban is
- Bináris file csak bináris módon nyitható meg

(Különbség az adatok tárolásának módjában van!!)

# File kezelés C-ben (7)

Használat módjának megadása:

”r” – létező file, olvasásra (ha nem létezik, hiba lép fel)

”w” – nem létező file létrehozására, és írására (ha létezik törli azt, és újra létrehozza!!)

”a” – létező file hozzáfűzésére (ha nem létezik, létrehozza)

”r+” – létező file olvasására, írására

”w+” – nem létező file létrehozására, írására, olvasására

”a+” – létező file hozzáfűzésére, írására, olvasására

# File kezelés C-ben (8)

## Filekezelő függvények:

- fopen() – megnyitás
- fclose() – lezárás
- fgetc(), fgets(), fscanf(), fread() – olvasás
- fputc(), fputs(), fprintf(), fwrite() – írás
- fseek() – pozicionálás a file-on belül (pointer állítása)
- ftell() – pointer állásának lekérdezése
- rewind() – file elejére való pozicionálás
- feof() – file végén van-e a file pointer

# File kezelés C-ben (9)

## Filekezelő függvények:

- `ferror()` – hibaállapot lekérdezése
- `fflush()` – memóriában levő puffer kiürítése írás/olvasás esetén
- `setbuf()` – memória puffer megadása (ha nem az automatikust akarjuk használni)
- `freopen()` – a file átirányítása

# File kezelés C-ben (10)

Mivel az írás, olvasás pufferen keresztül történik, lezárás előtt mindenképp javasolt az `fflush()`-sel történő puffer ürítése.

Írás, olvasása esetén a file pointer mindig automatikusan továbblép eggyel, a következő adatra.

# File kezelés C-ben (11)

Példa megnyitásra (1):

...

```
FILE * fp;
```

```
fp = fopen("adat.txt", "rt");
```

```
if (fp == NULL)
```

```
{
```

```
    printf("A file-t nem sikerült megnyitni!\n");
```

```
    return 1;
```

```
}
```

# File kezelés C-ben (12)

Példa megnyitásra (2):

```
...  
FILE * fp;  
if ((fp = fopen("adat.txt", "rt"))== NULL)  
{  
    printf("A file-t nem sikerült megnyitni!\n");  
    return 1;  
}  
...
```



# File kezelés C-ben (13)

Példa megnyitásra Windows alatt(3):

...

```
FILE * fp;
```

```
if ((fp = fopen("c:\\adatok\\adat.txt", "rt"))== NULL)
```

```
{
```

```
    printf("A file-t nem sikerült megnyitni!\n");
```

```
    return 1;
```

```
}
```

...

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i;
    FILE * fp;
    if ((fp = fopen ("c:\\adatok\\adat.txt", "wt")) == NULL)
    {
        printf("Hiba a megnyitaskor!\n");
        return 1;
    }
    for (i = 1; i <= 10; i++)
        fprintf(fp, "%3d", i);
    fflush(fp);
    fclose(fp);
```

// következő slide-on a visszaolvasás

```
if ((fp = fopen ("c:\\adatok\\adat.txt", "rt")) == NULL)
{
    printf("Hiba a megnyitaskor!\n");
    return 1;
}
while(!feof(fp))
{
    fscanf(fp, "%d", &i);
    printf("%d ", i);
}
fclose(fp);
return 0;
}
```

# File kezelés C-ben (14)

Az fwrite() használata:

```
fwrite(-----, -----, -----, -----);
```

Az 1. paraméter: a memória terület címe, ahonnan adatokat akarok kiírni.

A 2. paraméter: 1 adatelem mérete byte-okban (sizeof-fal)

A 3. paraméter: az adatelemek darabszáma

A 4. paraméter: a filepointer

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int t[10], i, db;
    FILE * fp;
    for (i = 0; i < 10; i++)
        t[i] = i+1;
    if ((fp = fopen ("adat.bin", "wb+")) == NULL)
    {
        printf("Hiba a megnyitaskor!\n");
        return 1;
    }
    fwrite(t, sizeof (int), 10, fp);
    fflush(fp);

    // következő slide-on a visszaolvasás
```

```
rewind(fp);  
db = fread(&i, sizeof(int), 1, fp);  
printf("%d ", i);  
fclose(fp);  
return 0;  
}
```

# Keresések

## Általános feltétel:

N rekordból álló halmazból a rekordok egyikének lokalizálása.

## További feltétel:

Minden rekord tartalmazzon egy kulcsmezőt.

## Feladat:

Megtalálni azt a rekordot, amelynek kulcsa megegyezik a keresett kulccsal.

## Két eset:

A keresés sikeres vagy sikertelen lehet.

# Szekvenciális keresések

1. Adott az  $R_1, R_2, \dots, R_n$  rekordok halmaza, ahol  $K_1, K_2, \dots, K_n$  jelöli a megfelelő kulcsokat.

Feltétel:  $n \geq 1$

## Megoldás:

az első rekord kulcsának összehasonlítása a keresett kulccsal.  
Ha megegyezik, a keresésnek sikeresen vége. Ha nem egyezik meg, vizsgálat a file végére. Ha vége, a keresésnek sikertelenül van vége. Ha nincs vége, akkor a következő rekord kulcsának a vizsgálata...



# Szekvenciális keresések (2)

## 2. Gyors szekvenciális keresés:

**Elv:** az előző algoritmusban 2 vizsgálat történik:

- kulcs egyezés van-e?
- file vége van-e?

Ha az egyik feltétel elvethető, akkor a keresés felgyorsul.

**Megoldás:** egy fiktív (ál)rekord a file végére  $R_{n+1}$ -ikként, amelyre nézve  $K_{n+1} = K$ . Így a file vége vizsgálat kimaradhat.

# Szekvenciális keresések (3)

## 3. Javított gyors szekvenciális keresés:

**Elv:** Megpróbálni egy cikluslépésen belül két rekordot megvizsgálni.

**Megoldás:** továbbra is a fiktív rekordot felvenni  $R_{n+1}$  rekordként, majd a ciklust  $i = -1$ -ről indítva, és a ciklusváltozót kétszer növelve ( $\text{Inc}(i); \text{Inc}(i);$ ) összehasonlítani az  $R_i, R_{i+1}$  rekordok  $K_i, K_{i+1}$  kulcsait  $K$ -val.

**Eredmény:** az 1. algoritmushoz képest kb. 30 %-kal gyorsabb!

# Szekvenciális keresések (4)

## 4. Rendezett táblában szekvenciálisan keresni:

**Elv:** Az előző algoritmusok csak akkor tudták eldönteni a sikertelen keresést, ha a file végére értek. Egy rendezett táblában ha a  $K_j$  kulcs nagyobb  $K$ -nál, akkor belátható, hogy a keresett rekord nincs a táblában.

**Eredmény:** sikertelen keresés esetén átlagosan kétszer gyorsabb!

# Szekvenciális keresések (5)

5. A tábla rendezettségi szempontját megváltoztatni:

**Elv:** feltehető, hogy a kulcsokra nem egyforma gyakorisággal hivatkozunk. Ekkor:

$K_i$  kulcs  $p_i$  valószínűséggel fordul elő, ahol:

$$p_1 + p_2 + \dots + p_n = 1 \quad (100 \%)$$

**Megoldás:**

Azokat a rekordokat előre tenni a táblában, melyeknek a gyakorisága nagyobb.

# Rendezések

Javasolt irodalom:

D. E. Knuth: A számítógép programozás művészete  
III. kötet

# Rendezések

## Alapelvek:

- Beszúró rendezés: egyesével tekinti a rendezendő számokat, és mindegyiket beszúrja a már rendezett elemek közé. (Kártyalapok rendezése)
- Cserélő rendezés: ha két elem nem a megfelelő sorrendben követi egymást, akkor felcserélésre kerülnek. Ez az eljárás ismétlődik mindaddig, míg további cserére már nincs szükség.
- Kiválasztó rendezés: először a legkisebb (vagy legnagyobb) elemet határozzuk meg, és a többitől valahogy elkülönítjük. Majd a következő legkisebbet választja ki, stb...
- Leszámoló rendezés: minden elemet összehasonlítunk minden elemmel. Az adott elem végső helyét a nála kisebb elemek száma határozza meg.

# Algoritmusok

- Leszámoló: -
- Beszűrő:
  - közvetlen beszűrás
  - bináris beszűrás
  - Shell rendezés
  - lista beszűrő rendezése
  - címszámító rendezés
- Cserélő:
  - buborék rendezés
  - Batcher párhuzamos módszere
  - gyorsrendezés (Quick sort)
  - számjegyes cserélés
  - aszimptotikus módszerek
- Kiválasztó:
  - közvetlen kiválasztás finomítása
  - elágazva kiválasztó rendezés
  - kupacrendezés
  - „legnagyobb be, első ki”
  - összefésülő rendezés
  - szétosztó rendezés

# Általános cél

Adott  $R_1, R_2, \dots, R_n$  rekordokat  $K_1, K_2, \dots, K_n$  kulcsaik nem csökkenő sorrendjébe kell rendezni lényegében egy olyan  $p(1), p(2), \dots, p(n)$  permutáció megkeresésével, amelyre

$$K_{p(1)} \leq K_{p(2)} \leq \dots \leq K_{p(n)}$$

fennáll.

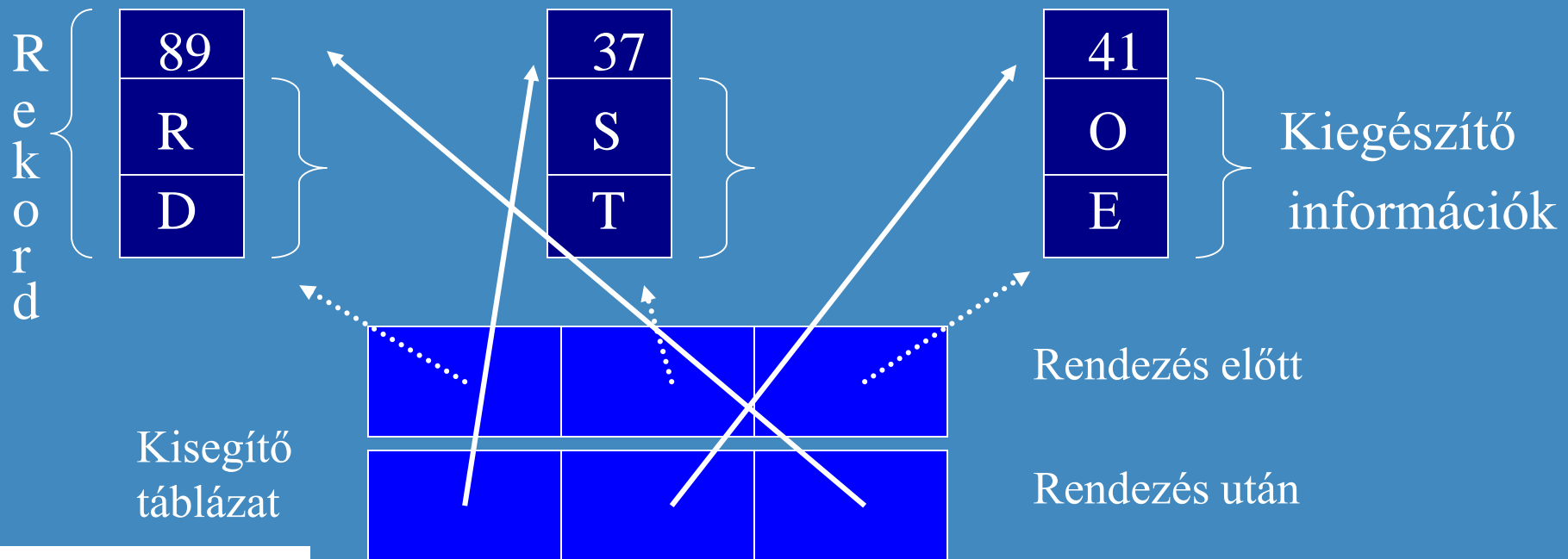
Ideiglenes feltétel:

Tekintsünk olyan halmazt, amelynek rendezése a memóriában megvalósítható.



# Célszerű adatszerkezetek

1. Ha a rekordok mindegyike több szót foglal el a tárban, akkor célszerű a rekordokra mutató láncolt címek új táblázatának létrehozása és ezeknek a kezelése a rekordok mozgatása helyett. Ez a **CÍMTÁBLÁZATOK** rendezése.



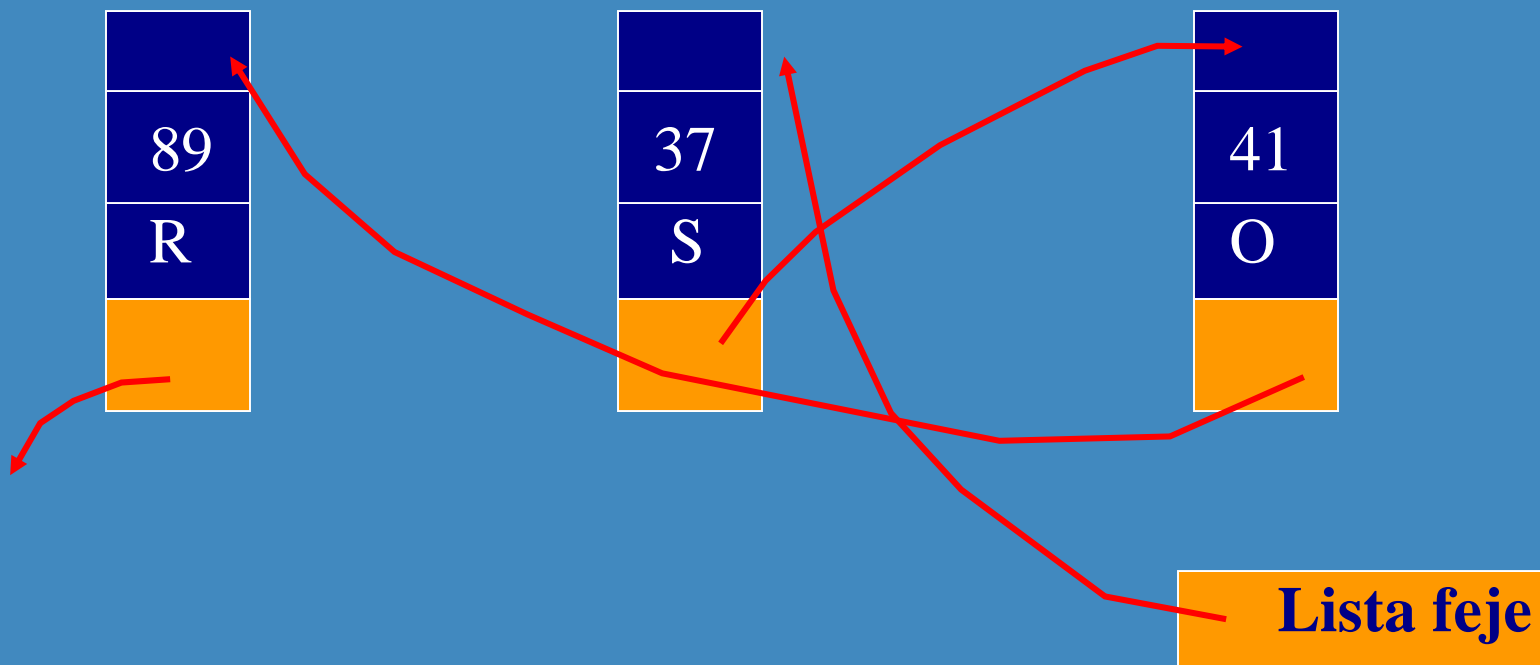
VBA Programozás

# Célszerű adatszerkezetek (2)

2. Ha a kulcs rövid, de a rekord kiegészítő információja hosszú, akkor a nagyobb sebesség elérése érdekében a kulcs a láncoló címmel együtt tárolható. Ez a **KULCSOK rendezése**.

# Célszerű adatszerkezetek (3)

3. Ha a láncoláshoz az egyes rekordokhoz csatolt segédmezőt alkalmaznak úgy, hogy a rekordok együtt egy lineáris listát alkotnak, amelyben minden láncoló cím a következő rekordra mutat, akkor az a **LISTA** rendezése.



# Leszámoló rendezés

**Elve:** a rendezett listában a  $j$ -ik kulcs pontosan  $j-1$  kulcsnál lesz nagyobb. (Ezért ha egy kulcsról tudjuk, hogy 27 másikonál nagyobb, akkor a neki megfelelő rekord sorszáma a rendezés után a 28 lesz.)

A kulcsokat minden lehetséges párosításban össze kell hasonlítani, és megszámolni, hogy az éppen vizsgált kulcsnál hány kisebb van. Variációk:

1. Hasonlítsuk össze  $K_j$ -t  $K_i$ -vel  $1 \leq j \leq N$  esetén, ha  $1 \leq i \leq N$

de felesleges:  $\begin{cases} \text{minden számot önmagával is összehasonlítani} \\ K_a\text{-t } K_b\text{-vel, majd utána } K_b\text{-t } K_a\text{-val összehasonlítani} \end{cases}$

2. Hasonlítsuk össze  $K_j$ -t  $K_i$ -vel  $1 \leq j \leq i$  esetén, ha  $1 \leq i \leq N$

Fontos! Az algoritmus nem jár együtt a rekordok mozgatásával, inkább a címtáblázat-rendezéshez hasonlít. Futási ideje:  $13N + 6A + 5B - 4$ , ahol:

$N$  = a rekordok száma,  $A = \begin{bmatrix} N \\ 2 \end{bmatrix}$ ,  $B$  az olyan indexpárok száma, amelyekre  $j < i$ , és  $K_j > K_i$

VBA Programozás

# Beszűrő rendezés (1)

(„Bridzsező módszer”)

Feltétel: Az  $R_j$  rekord vizsgálata előtt a megelőző  $R_1, \dots, R_{j-1}$  rekordok már rendezettek. Ekkor az  $R_j$  rekordot beszűrjük az előzőleg már rendezett rekordok közé. Változatok:

Közvetlen beszűrás, v. szitáló technika:

Tegyük fel, hogy  $1 < j \leq N$ , és hogy  $R_1, \dots, R_j$  rekordok rendezettek úgy, hogy  $K_1 \leq K_2 \leq \dots \leq K_{j-1}$ . Ekkor a  $K_j$  kulcsot összehasonlítjuk a  $K_{j-1}$ , majd a  $K_{j-2}$  kulcsokkal mindaddig, míg  $R_j$  be nem szűrhető  $R_i$  és  $R_{i+1}$  közé.

Bináris beszűrás:

Nem sorosan, hanem binárisan kell megkeresni az  $R_j$  rekord helyét a már rendezett rekordok között. **( $N > 128$  esetén már (!) nem javasolt)**

*Előnye:* kevesebb összehasonlítás

*Hátránya:* megtalált hely esetén továbbra is nagy mennyiségű rekordot kell megmozgatni a beszűráshoz.

# Beszűrő rendezés (2)

## Kétirányú beszűrés:

Bináris kereséssel a hely megkeresése, majd beszűrésnél abba az irányba mozgatjuk a már rendezett rekordokat, amerre kevesebbet kell mozgatni. (Jellemzője a nagyobb memória igény!).

## SHELL (v. fogó növekményes) rendezés:

Eddig a rekordok rendezéskor rövid lépésekkel kerültek helyükre. Most: rövid lépések helyett hosszú ugrások legyenek. Megalkotója: Donald L. Shell. (1959. július).

**Elve:** A rekordokat először (pl.) kettesével csoportokba osztjuk.

Ezeket a rekord csoportokat rendezzük (a két elem vagy jó sorrendben követi egymást, vagy felcseréljük őket). Ezután négyesével képezzük a csoportokat és rendezzük, majd újabb csoportokat képzünk, rendezzük, ..., míg a végén már csak egy csoport lesz, a teljes rekordhalmaz. Addigra az már rendezett részhalmazokból áll.

# Beszűrő rendezés (3)

Shell rendezés folyt: minden közbenső fázisra igaz, hogy **vagy viszonylag kicsi a csoport, vagy viszonylag jól rendezett**, ezért a rekordok gyorsan mozognak végső helyük felé.

Hatékonysága nagy mértékben függ az alkalmazott növekmények sorozatától. Erre táblázatban található meg a javasolt növekménysorozatok. Ha ez nincs kéznél, akkor:

$$n_1=1, \dots, n_{s+1} = 3 \times n_s + 1 \quad \text{és akkor legyen vége, ha:} \\ n_{s+2} > N$$

# Lista beszűrő rendezése

(A közvetlen beszűrás javítása.)

Az alapalgorithmus 2 alapvető műveletet alkalmaz:

- rendezett file átnézése adott kulcsnál kisebb vagy egyenlő kulcs megtalálása céljából.
- új rekord beszúrása a rendezett file meghatározott helyére

Kérdés: melyik az az adatszerkezet, amely erre a legalkalmasabb?

A file egy lineáris lista, amelyet ez az algorithmus szekvenciálisan kezel, ezért minden beszűrési művelethez átlagosan a rekordok felét kell megmozgatni.

Beszűráshoz az ideális adatszerkezet a láncolt lista, mivel csak néhány címet kell átírni a beszűráshoz. Ehhez elegendő az **egyirányú láncolt lineáris lista**.

Várható futási ideje:  **$7B + 14N - 3A - 6$**  időegység, ahol  $N$  = a rekordok száma

$A$  = a jobbról-balra maximumok száma,  $B$  = az eredeti permutációban levő inverziók száma (kb.  $1/4 N^2$ )



# Címszámító rendezés

**Elve:** könyvek vannak a padlón. Ezeket kell „abc” sorrendben egy polcra helyezni.

Meg kell becsülni a könyv végső helyét. Ezzel lecsökkenthető az elvégzendő összehasonlítások száma, valamint a rekord beszúrásához szükséges mozgatók száma.

**Hátránya:** általában  $N$ -nel arányos további tárterületet igényel azért, hogy a szükséges mozgatók száma lecsökkenjen.

# Cserélő rendezések

## Buborék rendezés:

Elve: hasonlítsuk össze  $K_1$ ,  $K_2$  kulcsokat. Helytelen sorrend esetén cseréljük fel  $R_1$ ,  $R_2$  rekordokat, aztán ugyanezt hajtsuk végre  $R_2$ ,  $R_3$ , majd  $R_3$ ,  $R_4$ ,  $R_4$ ,  $R_5$ , ... rekordokkal.

Az eljárás az  $R_N$ ,  $R_{N-1}$ ,  $R_{N-2}$ , ... rekordokat juttatja a megfelelő helyre.

Függőleges ábrázolás esetén látható, hogy a nagy számok fognak először „felbuborékolni” a helyükre.

Más neve: **cserélve kiválasztás v. terjesztés.**

Futási ideje:  $8A + 7B + 8C + 1$ , ahol

$A$  = a menetek száma,  $B$  = a cserék száma,

$C$  = az összehasonlítások száma

Max értéke:  $7.5 N^2 + 0.5 N + 1$  ( $N^2$  -tel arányos idő...)

# A buborék rendezés algoritmus

(Nem azért mert jó, hanem mert elterjedt...)

## Lépések:

1. Legyen korlát = N (korlát a legnagyobb index, amelyhez tartozó rekordról nem tudjuk, végső helyén van-e).
2.  $t = 0$ . Végezzük el a 3. lépést a  $j = 1, 2, \dots, \text{korlát} - 1$  értékekre, aztán menjünk a 4. lépésre. (Ha korlát = 1, akkor közvetlenül a 4. lépésre)
3.  $R_j$  és  $R_{j+1}$  összehasonlítása. Ha  $K_j > K_{j+1}$ , akkor cseréljük fel  $R_j$ -t  $R_{j+1}$ -gyel, és legyen  $t = j$ . (Ez jelzi, hogy volt csere...)
4. Történt-e csere? Ha  $t = 0$ , akkor nem, és az algoritmusnak így vége. Egyébként korlát =  $t$ , és visszatérni a 2. lépésre.

**(A közvetlen beszúrással összehasonlítva, kétszer lassúbb!!)**

# Batcher párhuzamos módszere

(Leginkább a Shell-rendezéshez hasonlítható.)

Elve: összehasonlításra ne szomszédos párokat válasszunk ki, hanem általánosan felírva:

$K_{i+1}$  -et  $K_{i+d+1}$  -gyel,

ahol  $d$  meghatározására van(!) algoritmus.

# Kiválasztó rendezés

## Elve:

1. Keressük meg a legkisebb kulcsot. A neki megfelelő rekordot tegyük a kimenő területre, majd helyettesítsük kulcsát végtelennel.
2. Ismételjük meg az 1. lépést. Ekkor a 2. legkisebb kulcsot találjuk meg, mivel az elsőt végtelennel helyettesítettük.
3. Az 1. lépést ismételni mindaddig, míg az összes  $N$  rekord kiválasztásra nem került.

Feltétel: a rendezés megkezdése előtt minden elem jelen legyen.

## Hátránya:

- a végső sorozatot egyesével generálja
- minden egyes elem kiválasztásához  $N-1$  összehasonlítást végez el.
- külön output területet igényel.

# Nem vesszük (sajnos...)

- gyorsrendezés (Quicksort)
- kupacrendezés
- „legnagyobb be, első ki”
- összefésülés
- szétosztó rendezés

(De aki gondolja, utánanézhethet)

Házi feladat:

Tessék megpróbálni megkeresni a cserélve történő kiválasztás helyét a tanult rendezési alapelvek között! (vajon cserélő, vagy kiválasztó-e?)

# Újabb program: kör (K=, T=)

cmdKerulet

cmdTerulet

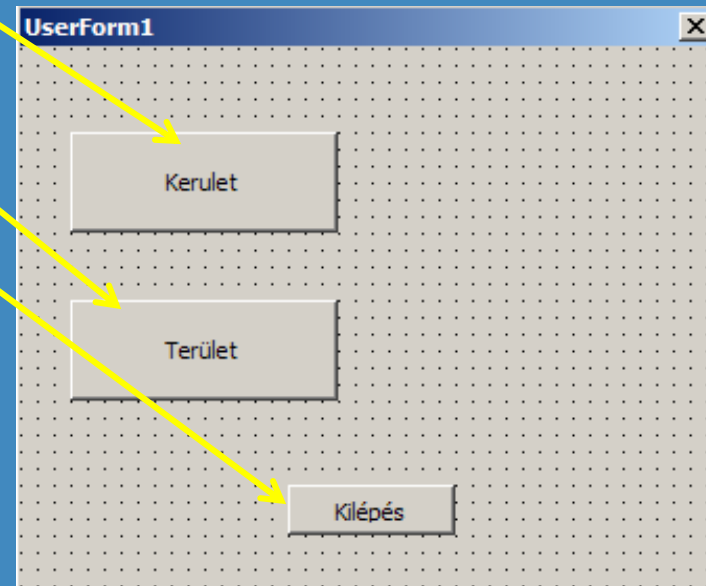
cmdKilepes

A Caption rendre:

Kerület

Terület

Kilépés



# TextBox

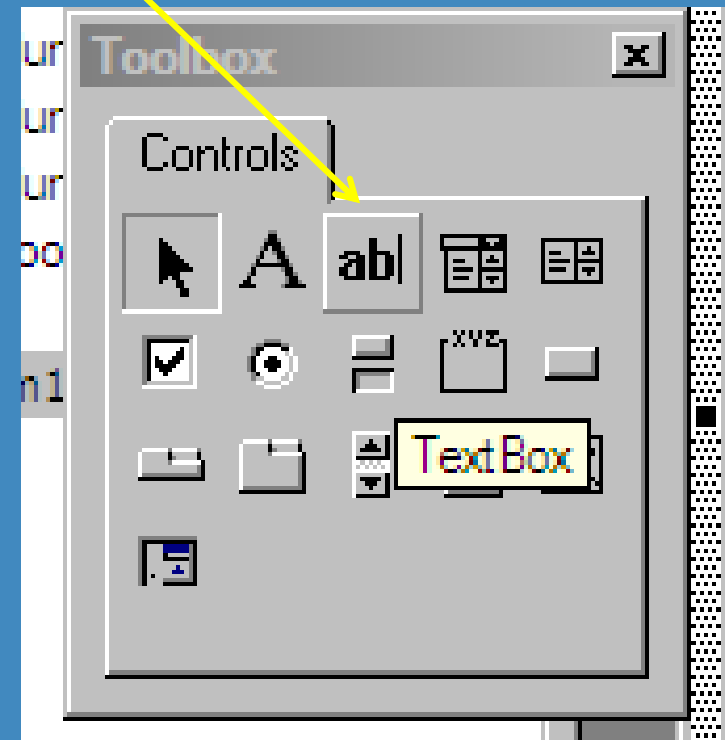
Kijelölni a Toolbox-on a TextBox-ot,  
Majd rajzolni egy-egy TextBox-ot  
a Kerület és a Terület mellé.

Nevük:

txtKerulet

txtTerulet

Ide fog az eredmény kerülni.





# Síkdomok

Elkészíteni a cmdKerulet, cmdTerulet  
parancsgombok

Click eseményét

A kör sugarát az A1  
cellába fogjuk írni

Hivatkozás: Cells(1,1)

Pl.:

$\text{txtKerulet} = \text{Cells}(1,1) * 2 * 3,1415$

# Kilépés

cmdKilepes click eseménye: End

```
Private Sub cmdKerulet_Click()  
    txtKerulet = Cells(1, 1) * 2 * 3.1415  
End Sub
```

```
Private Sub cmdKilepes_Click()  
    End  
End Sub
```

```
Private Sub cmdTerulet_Click()  
    txtTerulet = Cells(1, 1) * Cells(1, 1) * 3.1415  
End Sub
```

# Futtatás

- Előtte az A1 cellába be kell írni a kör sugarát, pl.: 5

The screenshot shows an Excel spreadsheet with the following data:

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

The UserForm1 is open, displaying the following information:

- Kerület** (Perimeter): 31.415
- Terület** (Area): 78.5375
- Kilépés** (Exit) button

VBA

# Változók

- Skalár változók
- Tömb változók

(Az egyszerűség kedvéért: változó és tömb)

Pl.: `txtKerulet = Cells(1,1) * 2 * 3,1415` helyett

`K = Cells(1,1) * 2 * 3,1415`

Itt K változó (nem vezérlő)

# Feladat

- Excel indítása után új UserForm
- 2 parancsgomb: cmdValtozok, cmdKilepes  
Caption: Változók, Kilépés
- 5 TextBox:  
txtByte, txtInteger, txtLong, txtDouble, txtString

A TextBox nevének nem kell igazodnia a beleírt érték típusához, sem tartalmához, de javasolt!

Most a tárolni kívánt érték típusa alapján történik.

# Feladat (folyt)

- cmdValtozok click eseménye:
  - Dim a As Byte
  - Dim b As Integer
  - Dim c As Long
  - Dim d As Double
  - Dim e As String

# Feladat (folyt)

- A cmdValtozok click esemény kódjának folytatása
  - $A = 100$
  - $B = 1000$
  - $C = 10000$
  - $D = 3.1415926535$
  - $E = \text{„Abraka-dabra”}$

# Feladat (folyt)

- A cmdValtozok click esemény kódjának folytatása
  - txtByte = a
  - txtInteger = b
  - txtLong = c
  - txtDouble = d
  - txtString = e
- A cmdKilepes click eseményéhez:
  - End



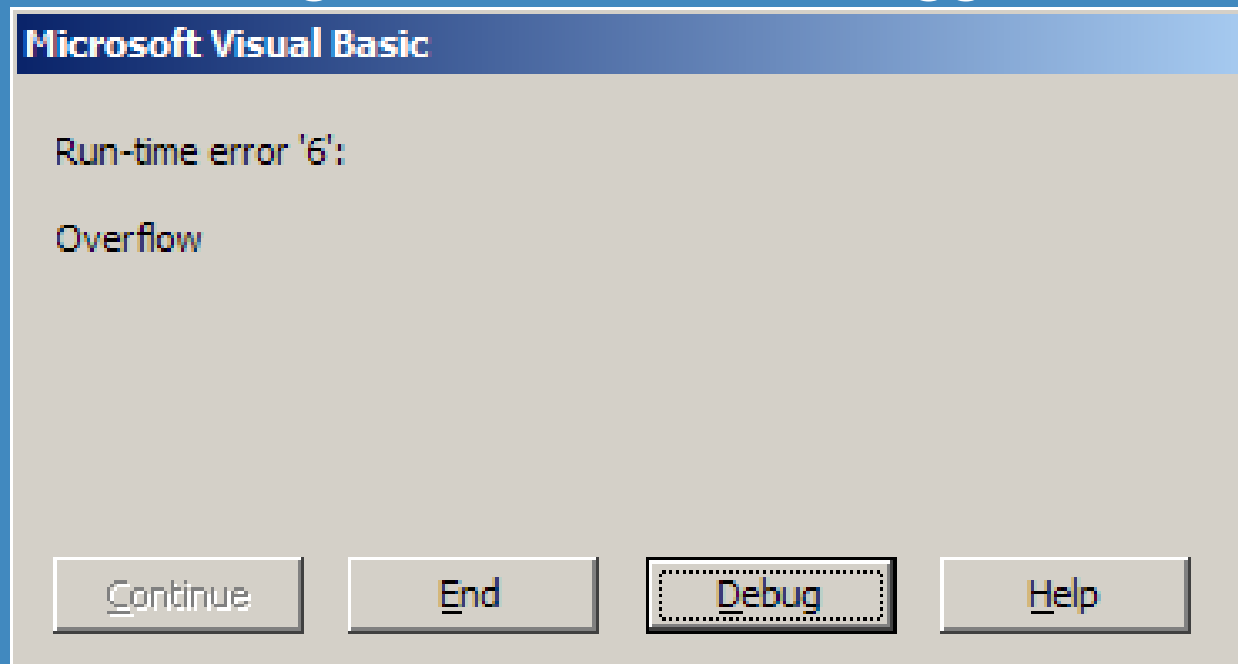
# Feladat futtatása

UserForm1

Változók	100
Kilépés	1000
	100000
	3.1415926535
	Abraka-dabra

# Nem megfelelő adat esetén

- Pl.: Byte típus esetén  $a = 300$  nem értelmes (maximális értéke 255 lehet)
- Ekkor hiba lép fel
- De tessék kipróbálni negatív értékkel, szöveggel, ...



# Új feladat

## (adatbeolvasás textbox-ból)

Készítsünk egy olyan programot, amelyik 3 TextBox-szal rendelkezik.

Az elsőbe majd beírjuk, hogy hány darab almánk van, a másodikba, hogy hány felé kell osztani, a harmadikba az eredményt várjuk.

Készítsük el ezt a programot.

# Tennivalók

- Elindítani az Excel-t, majd a fejlesztőrendszert valamelyik tanult módon.
- Létre hozni egy UserForm-ot.
- Elhelyezni rajta két parancsgombot. Az első neve legyen cmdSzamol, felirata Számol, a másodiké cmdKilepes, felirata Kilépés.
- Elhelyezni rajta 3 db TextBox-ot. Nevük sorra txtAlma, txtLetszam, txtEredmeny.

# A form kinézete

- txtAlma
- txtLetszam
- cmdSzamol
- txtEredmeny
- cmdKilepes

The image shows a VBA UserForm titled "UserForm1" with a standard Windows-style title bar (minimize, maximize, close buttons). The form has a light gray background with a dotted grid pattern. It contains five controls arranged vertically on the right side:

- A text box (txtAlma) at the top.
- A second text box (txtLetszam) below the first.
- A command button (cmdSzamol) labeled "Számol" below the second text box.
- A third text box (txtEredmeny) below the command button.
- A command button (cmdKilepes) labeled "Kilépés" at the bottom.

Five yellow arrows originate from the list on the left and point to each of these controls in sequence.

# Feliratok

- lblAlma
- lblLetszam
- lblEredmeny

Almák száma:

Létszám:

Eredmény:

Számol

Kilépés

# Kód

- A cmdKilepes click eseménye a szokásos: End
- A cmdSzamol click eseménye:  
Dim Alma As Integer  
Dim Letszam As Integer  
Dim Eredmeny As Integer  
Alma = txtAlma  
Letszam = txtLetszam  
Eredmeny = Alma / Letszam  
txtEredmeny = Eredmeny

# Futtatás

1. Almák száma = 6

Létszám = 2

Eredmény = ? (3)

2. Almák száma = 5

Létszám = 2

Eredmény = ? (2)

Eltűnt 1 alma! De hova??

Változtassuk meg Eredmeny típusát Double-ra, és ...



# Következtetés

- Már a program írásakor meg kell tudnom becsülni, hogy ez egyes változóknak milyen értéket kell tudniuk tárolni.
- Lesz osztás?
  - Lehet, hogy nullával fogok osztani?
- Lesz benne négyzetgyökvonás?
  - Lehet, hogy negatív számból fogok gyököt vonni?

# Önálló program készítése

- Excel indítása
- Az első oszlopba felvinni 5 gyerek nevét
- A második oszlopba felvinni a testsúlyukat
- Programot írni, amely két parancsgomb, és két textbox segítségével meghatározza és kiírja a gyerekek összsúlyát, illetve átlagsúlyát.
- Az Excel táblába, szövegesen

Pl.: `Cells(8,1) = „Átlagsúly”`

`Cells (8,2) = OsszSuly / 5`

Legyen egy harmadik nyomógomb is a kilépéshez.

# Önálló program készítése

- Új Excel indítása
- Az első oszlopba felvinni 5 termék nevét
- A második oszlopba felvinni a nettó árakat
- Programot írni, amely egy parancsgomb segítségével meghatározza és kiírja a következő oszlopba a bruttó árat

Legyen egy második nyomógomb is a kilépéshez.

# Önálló feladat

- Elindítani az Excel-t, majd a fejlesztőrendszert valamelyik tanult módon.
- Létrehozni egy UserForm-ot.
- Elhelyezni rajta két parancsgombot. Az első neve legyen cmdFuttatas, felirata Futtat, a másodiké cmdKilepes, felirata Kilépés.
- A Kilépés gomb click eseményét elkészíteni az előző programokhoz hasonlóan.

# Önálló feladat

Option Base 1

Private Type Ember

VezetekNev As String

KeresztNev As String

Fizetes As Long

GyerekekSzama As Byte

End Type

# Önálló feladat

A cmdFuttatas click eseményére:

Dim Dolgozo As Ember

Dolgozo.VezetekNev = "Kovács"

Dolgozo.KeresztNev = "János"

Dolgozo.Fizetes = 120000

Dolgozo.GyerekekSzama = 2

MsgBox (Dolgozo.VezetekNev & " " & Dolgozo.KeresztNev)

# Új önálló feladat

- Vagy új Excel-lel, vagy a meglévő program tovább írásával:
- Tömbváltozó használatára átírni. Az eredmények (nevek, fizetés, gyerekek száma) az Excel tábla első oszlopában jelenjenek meg.

Pl.: `Cells(1, 1) = Dolgozok(5).VezetekNev`

# Önálló feladat megoldása

	A	B	C	D	E	F	G
1	Kempelen						
2	Farkas						
3	160000						
4	0						
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							

UserForm1

Futtatás

Kilépés

Microsoft Excel

Kovács János

OK

```
Munkafüzet1 - UserForm1 (Code)

cmdKilepes

Option Base 1
Private Type Ember
    VezetekNev As String
    KeresztNev As String
    Fizetes As Long
    GyerekekSzama As Byte
End Type

Private Sub cmdFuttat_Click()
    Dim Dolgozo As Ember
    Dolgozo.VezetekNev = "Kovács"
    Dolgozo.KeresztNev = "János"
    Dolgozo.Fizetes = 120000
    Dolgozo.GyerekekSzama = 2

    MsgBox (Dolgozo.VezetekNev & " " & Dolgozo.KeresztNev)

    Dim Dolgozok(100) As Ember
    Dolgozok(5).VezetekNev = "Kempelen"
    Dolgozok(5).KeresztNev = "Farkas"
    Dolgozok(5).Fizetes = 160000
    Dolgozok(5).GyerekekSzama = 0

    Cells(1, 1) = Dolgozok(5).VezetekNev
    Cells(2, 1) = Dolgozok(5).KeresztNev
    Cells(3, 1) = Dolgozok(5).Fizetes
    Cells(4, 1) = Dolgozok(5).GyerekekSzama

End Sub

Private Sub cmdKilepes_Click()
    End
End Sub
```



# Az előző Dolgozo-s program folytatása

Legyen egy új  
parancsgomb  
(cmdCiklus)

Click eseménye  
pedig:

```
Private Sub cmdCiklus_Click()  
    Dim Dolgozok(10) As Ember  
    For i = 1 To 10  
        Dolgozok(i).VezetekNev = "Kempelen"  
        Dolgozok(i).KeresztNev = "Farkas"  
        Dolgozok(i).Fizetes = 160000 + i  
        Dolgozok(i).GyerekekSzama = i  
        Cells(1, i) = Dolgozok(i).VezetekNev  
        Cells(2, i) = Dolgozok(i).KeresztNev  
        Cells(3, i) = Dolgozok(i).Fizetes  
        Cells(4, i) = Dolgozok(i).GyerekekSzama  
    Next  
End Sub
```

# Önálló munka

Meghatározni a Fibonacci sorozatot első 10 tagját  
Excel tábla első oszlopába egymás alá.

1, 1, 2, 3, 5, 8, 13, 21, ...

Új form, (frmForFibo) két parancsgomb,  
(cmdSzamol, cmdKilep)

# Önálló feladat

- Tessék előállítani a Fibonacci sorozat első 10 tagját mind a 4 tanult Do ... Loop szerkezettel.
- Emlékeztetőül: 1, 1, 2, 3, 5, 8, 13, 21, ...
- Új form (frmLoopFibo), két parancsgomb (cmdSzamol, cmdKilep).

# Önálló feladat

- 4 Loop-os szerkezettel meghatározni, hány tagot kell összeadni a következő sorozatból, hogy elérjük a határt (S).

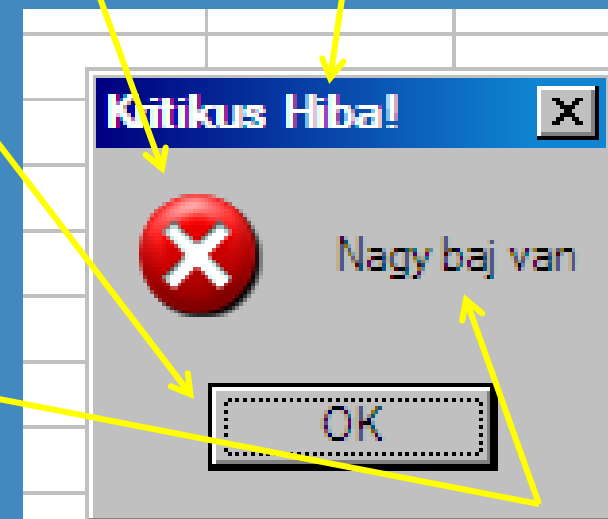
$$S = 1/1 + 1/2 + 1/3 + 1/4 + \dots + 1/n$$

- Határt (S) olvassa Cells(2, 2)-ből
- Eredményeket (n): Cells(3, 2), Cells(3, 3), Cells(3, 4), Cells(3, 5)
- Új form (frmReciprok), négy + 1 parancsgomb  
(cmdSzamol1, cmdSzamol2, cmdSzamol3, cmdSzamol4, cmdKilep)

# MsgBox paramétereit

MsgBox "Üzenet", gomb(ok) és jelleg, "Cím"

```
If (i > 5) And (i < 40) Then  
    MsgBox "Nagy baj van", vbOKOnly + vbCritical, "Kritikus Hiba!"  
End If
```

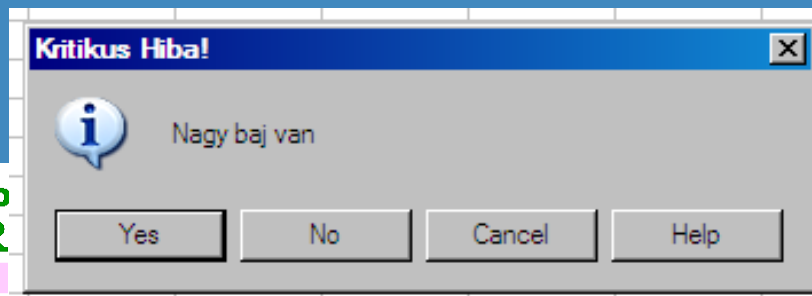
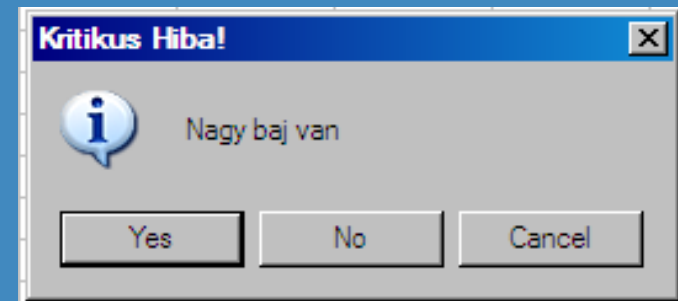
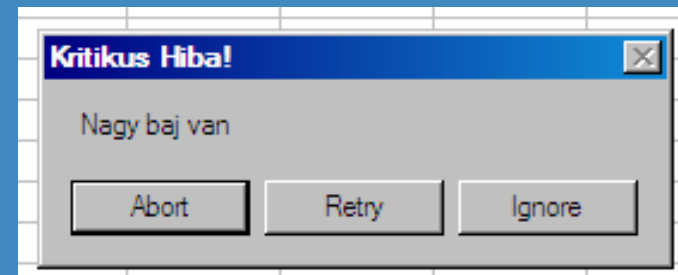
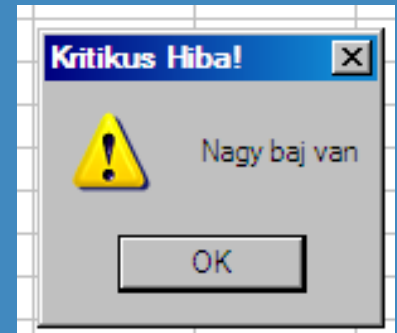
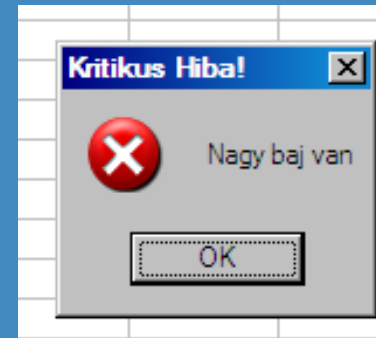


VBA Programozás

# MsgBox konstansok

A gomb(ok) és a jelleg **konstansokkal** vezérelhető. Pl.:

vbOKOnly.	0
vbOKCancel	1
vbAbortRetryIgnore	2
vbYesNoCancel	3
vbYesNo	4
vbRetryCancel	5
vbCritical	16
vbQuestion	32
vbExclamation	48
vbInformation	64



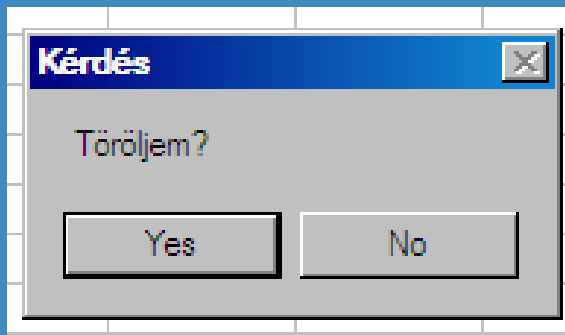
# MsgBox visszatérési érték

Lekérdezhető, melyik gombot nyomták meg.

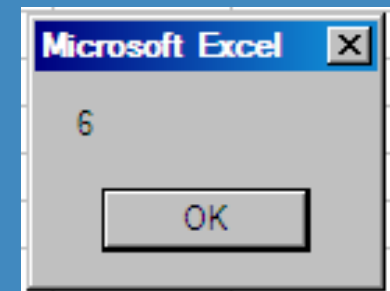
Ekkor azonban **másképp kell hívni!**

Eredm = MsgBox ("Töröljem?", vbYesNo, "Kérdés")

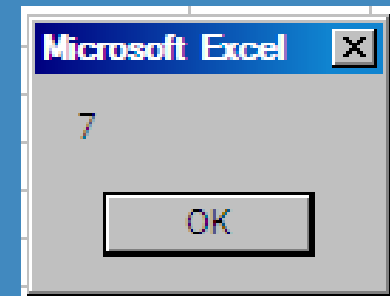
```
If (i > 5) And (i < 40) Then  
    Eredm = MsgBox("Töröljem?", vbYesNo, "Kérdés")  
    MsgBox (Eredm)  
End If
```



Yes



No



# MsgBox visszatérési érték

vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No



# Önálló feladat

1 és 100 közötti véletlenszerűen generált valós értékekkel feltölteni az Excel tábla első oszlopának első 30 elemét.

Megszámolni, ezek közül hány érték nagyobb, mint 50.

Az eredményt kiíratni MessageBox-ban OK gombbal.

Ehhez azonban kell a véletlenszám generálás...

# Véletlenszám generálás

- Az Rnd() függvény minden hívásakor vissza ad egy  $0 \leq x < 1$  valós értéket.
- Véletlenszerű, de ismétlődő (!) generálás.
- Emiatt van egy Randomize eljárás

# Véletlenszám generálás

- Tetszőleges intervallumból:

$$X = A_h + \text{Rnd}() * (F_h - A_h + 1)$$

Pl.: 3 és 10 közötti véletlenszám:

$$x = 3 + \text{Rnd}() * (10 - 3 + 1)$$

$$3 + 0 \dots 1 * 8 \rightarrow \underline{3 \leq x < 11}$$

If x > 10 Then x = 10

# Önálló feladat megoldása

	A	B	C	D	E	F
1	64.55709					
2	51.60881					
3	22.33232					
4	58.27582					
5	74.85327					
6	45.64704					
7	90.43737					
8	28.30148					
9	66.75384					
10	89.01273					
11	28.13814					
12	36.80561					
13	1.422113					
14	29.4659					
15	98.4046					
16	76.1283					
17	58.08784					
18	60.56734					
19	26.80438					
20	50.81665					
21	79.35966					
22	85.74113					
23	9.315258					
24	54.18735					
25	41.53764					
26	7.337105					
27	45.13767					
28	69.88905					
29	49.9257					
30	40.69341					

UserForm1

Random

Számlálás 16

Kilépés

```
Private Sub cmdKilepes_Click()  
    End  
End Sub
```

```
Private Sub cmdRandom_Click()  
    Dim i As Integer  
    For i = 1 To 30  
        Cells(i, 1) = Rnd() * 100  
    Next  
End Sub
```

```
Private Sub cmdSzamlalas_Click()  
    Dim Szamlal As Integer  
    Dim i As Integer  
    Szamlal = 0  
    For i = 1 To 30  
        If Cells(i, 1) > 50 Then  
            Szamlal = Szamlal + 1  
        End If  
    Next  
    txtSzamlalas = Szamlal  
End Sub
```

# Önálló feladat

- Új Excel, új UserForm.
- 4 parancsgomb: Sorsolás, Tippek, Találatok, Kilépés.
- Sorsolás: generálni 5 db egész számot az 1 ... 90. intervallumból úgy, hogy egy szám legfeljebb egyszer fordulhasson elő (lottó). Letárolni az első oszlopban egymás alá.
- Tippek: InputBox-szal beolvasni 5 db számot (tippek). Letárolni a második oszlopban. (Nem kell ellenőrizni)
- Találatok: meghatározni, hány találat van, és kiírni MessageBox-szal. OK gomb legyen rajta.
- Kilépés: kilép.

# Önálló feladat megoldása

UserForm 1

Sorsolás

Tippek

Találatok

Kilépés

	A
1	71
2	2
3	69
4	74
5	65
6	

	A	B
1	71	3
2	2	7
3	69	11
4	74	69
5	65	74
6		

Értékelés

Találatok száma:2

OK

# Önálló feladat megoldása

```
Private Sub cmdSorsolas_Click()  
    Dim Sorsolas(5) As Byte  
    Dim Van As Byte  
    Dim Jo As Boolean  
    Dim i, s As Integer  
  
    Sorsolas(1) = 1 + Rnd() * 90  
    If Sorsolas(1) > 90 Then  
        Sorsolas(1) = 90  
    End If  
    Van = 1  
    Do  
        s = 1 + Rnd() * 90  
        If s > 90 Then  
            s = 90  
        End If  
        Jo = True  
        i = 1  
        Do  
            If Sorsolas(i) = s Then  
                Jo = False  
            End If  
            i = i + 1  
        Loop Until i > Van  
        If Jo Then  
            Van = Van + 1  
            Sorsolas(Van) = s  
        End If  
    Loop Until Van = 5  
    For i = 1 To 5  
        Cells(i, 1) = Sorsolas(i)  
    Next i  
End Sub
```

```
Private Sub cmdTalalatok_Click()  
    Dim i, j As Byte  
    Dim Talalat As Byte  
    Talalat = 0  
    For i = 1 To 5  
        For j = 1 To 5  
            If Cells(i, 2) = Cells(j, 1) Then  
                Talalat = Talalat + 1  
            End If  
        Next j  
    Next i  
    MsgBox "Találatok száma:" & Talalat, vbOKOnly, "Értékelés"  
End Sub
```

```
Private Sub cmdTippek_Click()  
    Dim i As Byte  
  
    For i = 1 To 5  
        Cells(i, 2) = InputBox("Kérem adja meg " & i & " tippjét!")  
    Next i  
End Sub
```

# Önálló feladat

- A Lottó előtti programot folytatni úgy, hogy az 50-nél nagyobb elemek összege is kell.



# Önálló feladat megoldása

	A	B	C	D	E	F	G
1	70.55475						
2	53.3424						
3	57.95186						
4	28.95625						
5	30.1948						
6	77.47401						
7	1.401764						
8	76.07236						
9	81.44901						
10	70.90379						
11	4.535275						
12	41.40327						
13	86.26193						
14	79.048						
15	37.35362						
16	96.19531						
17	87.14458						
18	5.623686						
19	94.95567						
20	36.40187						
21	52.48684						
22	76.71117						
23	5.350452						
24	59.24583						
25	46.87001						
26	29.81654						
27	62.26967						
28	64.78212						
29	26.37929						
30	27.93421						
31							

UserForm1

Random

Összegzés

1569.07033514977

Kilépés

cmdRandom

Option Base 1  
Option Explicit

```
Private Sub cmdKilepes_Click()  
    End  
End Sub
```

```
Private Sub cmdOsszegzes_Click()  
    Dim Osszeg As Double  
    Dim i As Integer  
  
    Osszeg = 0  
    For i = 1 To 30  
        Osszeg = Osszeg + Cells(i, 1)  
    Next i  
    txtOsszeg = Osszeg  
End Sub
```

```
Private Sub cmdRandom_Click()  
    Dim i As Integer  
  
    Randomize  
    For i = 1 To 30  
        Cells(i, 1) = Rnd() * 100  
    Next i  
End Sub
```

# Önálló feladat

- Elvben (és gyakorlatban) a számok most rendezve vannak. Kell 2 új parancsgomb: cmdBeszur, cmdTorol
- **cmdBeszur**: InputBox segítségével olvassunk be egy számot a billentyűzetről, és szúrjuk be a megfelelő helyre a rendezettség megtartásával
- **cmdTorol**: InputBox segítségével olvassunk be egy számot a billentyűzetről, keressük meg a listában és töröljük ki.

Nem nehéz, de érdekes, és gondolkodtató...

# Varietas delectat

## (A változatosság gyönyörködtet)

A Variant típus

Dim a → a Variant lesz, mert nincs megadva típus

Dim a, b As Integer → b Integer lesz, a Variant

Dim a As Variant → a Variant lesz

# A Variant

Speciális típus. Bármilyen értéket képes tárolni.  
(Egy kivétellel: fix hosszúságú string)

a = 123

a = "123"

a = 123.234

# Konverzió (Conversion)

- CBool (kifejezés)
- CByte (kifejezés)
- CDate (kifejezés)
- CDb1 (kifejezés)
- CDec (kifejezés)
- CInt (kifejezés)
- CLng (kifejezés)
- CSng (kifejezés)

# CBool

```
Dim A As Byte, B As Byte  
Dim Check As Boolean
```

```
A = 5
```

```
Check = CBool(A)
```

```
B = 0
```

```
Check = CBool(B)
```

True

False

# CByte

```
Dim MyDouble As Double  
Dim MyByte As Byte  
MyDouble = 125.5678  
MyByte = CByte(MyDouble)
```

# CByte (For ciklus + Step)

- Step-pel meg lehet adni a lépésközt.
- A lépésköz lehet egész, lehet tört, lehet negatív

```
Dim i
Cells = Empty
For i = 1 To 100 Step 3.5
    Cells(CByte(i), 1) = i
Next i
```



	A	
1	1	
2		
3		
4	4.5	
5		
6		
7		
8	8	
9		
10		
11		



# CDate

```
Dim firstDate, secondDate As Date
Dim timeOnly, dateAndTime, noDate As String
Dim dateCheck As Boolean
firstDate = CDate("February 12, 1969")
secondDate = #2/12/1969#
timeOnly = "3:45 PM"
dateAndTime = "March 15, 1981 10:22 AM"
noDate = "Hello"
dateCheck = IsDate(firstDate) → True
dateCheck = IsDate(secondDate) → True
dateCheck = IsDate(timeOnly) → True
dateCheck = IsDate(dateAndTime) → True
dateCheck = IsDate(noDate) → False
```

# CCur, CSng, CDbl

Ccur: Currency-re konvertál

```
Dim a, b                                'Itt a is b is Variant
a = CCur(234.456784)                  'Itt az a Currency típusú
b = CDbl(a * 8.2 * 0.01)               'Itt a b Double típusú
```

CSng: Single-re konvertál

CDbl: Double konvertál

(mindkettő valós típus)

# CDec

```
Dim a, b
TextBox1.Text = "160000"
a = TextBox1.Text
b = CDec(a)
```

- Az a értéke: "160000" (String)
- A b értéke: 160000 (Decimal)

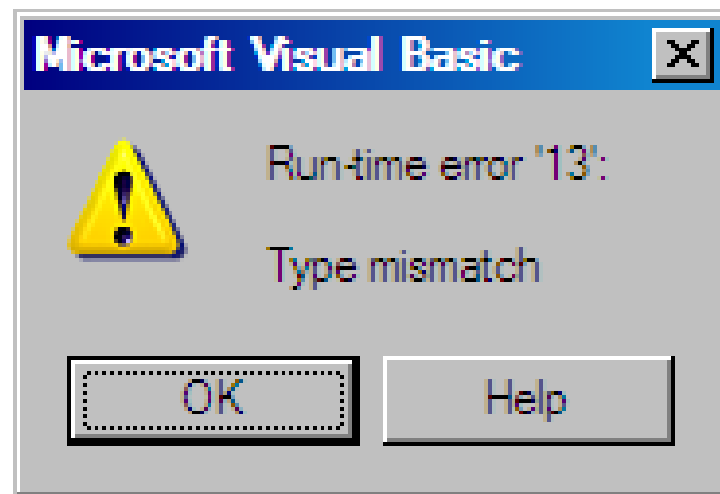
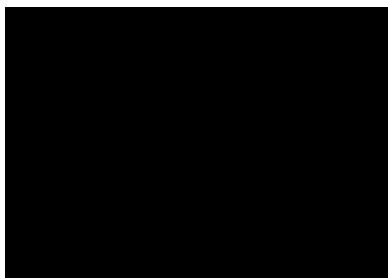
# CInt, CSng

```
Dim a, b
TextBox1.Text = "1600.76"
a = TextBox1.Text
b = CInt(a)
```

- Az a értéke "1600.76" (String)
- A b értéke 1601 (Integer)
- Kerekítés történt a konverzió közben !!

# Jó, konvertálunk... (jó?)

```
Dim a As Variant  
TextBox1.Text = "Mákos rétes"  
a = CInt(TextBox1.Text)
```

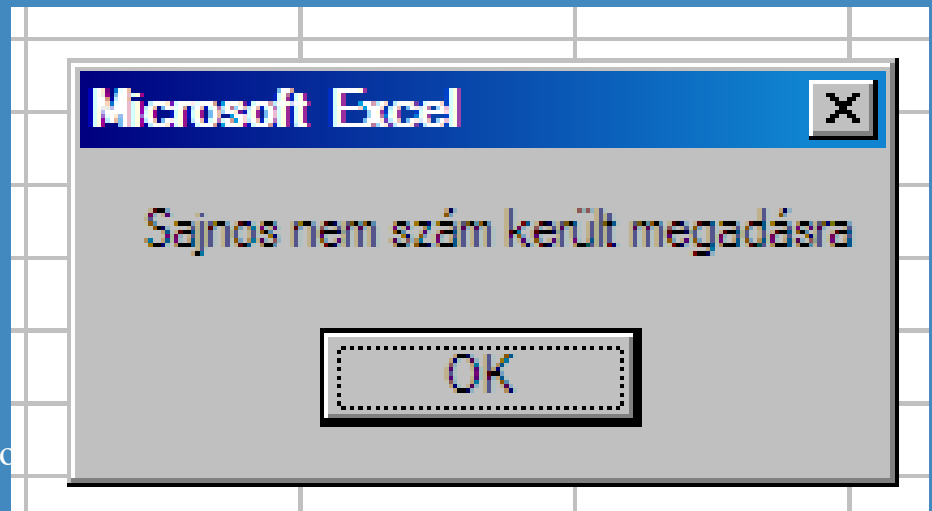


- De hisz nem is szám van a Textbox-ban!!!

# Konvertálás előtt ellenőrzés kell

```
Dim a As Variant
TextBox1.Text = "Mákos rétes"
If IsNumeric(TextBox1.Text) Then
    a = CInt(TextBox1.Text)
Else
    MsgBox ("Sajnos nem szám került megadásra")
End If
```

- IsNumeric !!



# IsNumeric()

- Variant típusú paramétert vár
  - Logikai értékkel tér vissza
  - Ha a Variant-ban numerikus jellegű érték van, akkor True értéket ad vissza
  - Egyébként False
- 
- Ha a paraméternek Date formátumú értéke van, az nem Numeric. Az eredmény tehát: False lesz

# IsArray()

- Paraméterének változónak kell lennie
- Logikai értékkel tér vissza
- Ha a változó tömbváltozó, akkor az eredmény True
- Egyébként False



# IsDate()

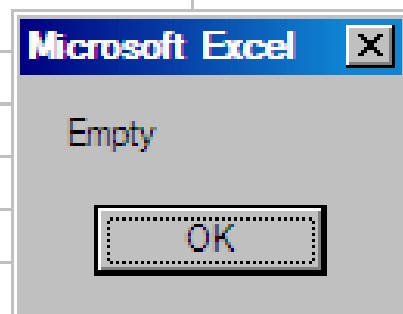
- Paraméterének Variant típusúnak kell lennie
- Ha a paraméternek dátum jellegű értéke van, akkor az eredmény True
- Egyébként False

# IsEmpty()

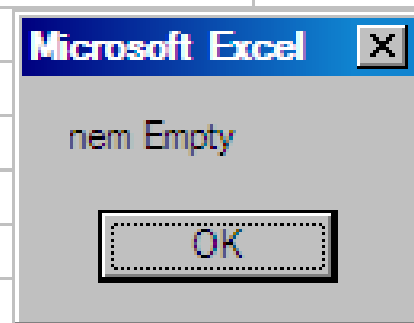
- Paraméterének Variant típusúnak kell lennie
- Ha a paraméter üres, akkor eredménye True
- Egyébként False

```
If IsEmpty(Cells(1, 1)) Then  
    MsgBox ("Empty")  
Else  
    MsgBox ("nem Empty")  
End If
```

	A	B	C
1			
2			
3			
4			
5			
6			
7			
8			



	A	B	C
1	0		
2			
3			
4			
5			
6			
7			
8			



VBA P

# Null, IsNull()

- Null érték: a Variant(!) típusú változó nem tartalmaz érvényes adatot
- IsNull()
- Variant típusú paramétert vár
- Ha értéke Null, akkor eredménye: True
- Egyébként False

# Null, IsNull()

!!!!

```
Dim a As Variant
a = Null
If IsNull(a) Then
    MsgBox ("Null")
Else
    MsgBox ("nem Null")
End If
```

```
Cells(1, 1) = Null
If IsNull(Cells(1, 1)) Then
    MsgBox ("Null")
Else
    MsgBox ("nem Null")
End If
```

Microsoft Excel

Null

OK

VBA Pro

Microsoft Excel

nem Null

OK

# IsMissing()

- Kicsit előre szaladunk, de mindjárt jön az anyagban.
- Egy eljárás (Sub) vagy függvény (Function) kaphat paramétereket.
- Ha egyes paraméterüket elhagyhatónak adtuk meg, és a hívásnál elhagytuk, akkor az

**IsMissing(paraméter neve)**

- Eredménye True
- Egyébként False

A paraméter típusa csak String vagy Variant lehet!!!

# UserForm\_Initialize()

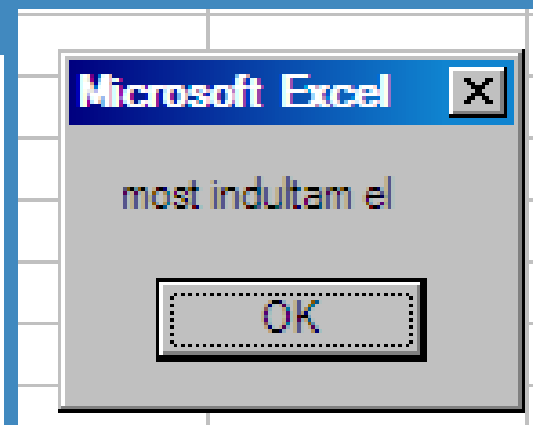
- A Userform betöltődésekor (induláskor) automatikusan végrehajtódik

```
Private Sub UserForm_Initialize()  
  
    MsgBox ("most indultam el")  
    'Ide lehet írni olyan utasításokat,  
    'amik előkészítik a program futását  
    'pl: cellákat törölnék, vagy épp  
    'cellákat töltenek fel  
    'pl.: töröljünk minden cellát
```

→ Cells = Empty

End Sub

Tanszék



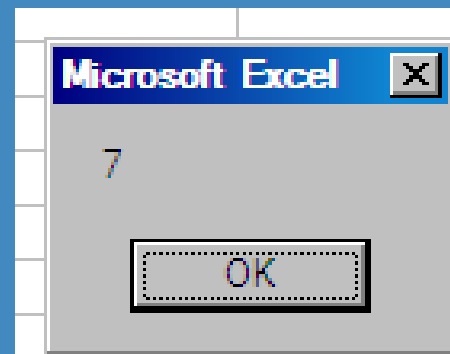
# Saját alprogram

- Lehet eljárás (Sub)
- Lehet függvény (Function)
- A függvénynek van eredménye, amit vissza ad
- Az eljárásnak nincs

# Függvény (Function)

- Paraméterlista: formális, aktuális
- Függvény meghívása
- Függvénynek van visszatérési értéke, és legyen is!
- Eredménye csak 1 van! Még két eredménye se lehet!

```
Private Sub UserForm_Initialize()  
    Dim x1 As Byte, x2 As Byte  
    x1 = 5  
    x2 = 7  
    MsgBox (Nagyobb(x1, x2))  
End Sub  
  
Function Nagyobb(a As Byte, b As Byte) As Byte  
    If a > b Then  
        Nagyobb = a  
    Else  
        Nagyobb = b  
    End If  
End Function
```





# Eljárás (Sub)

- Lehet paramétere
- Visszatérési értéke nincs
- Ha eredményt akarunk visszaadni, akkor azt csak paraméter(ek)ben tudjuk visszaadni

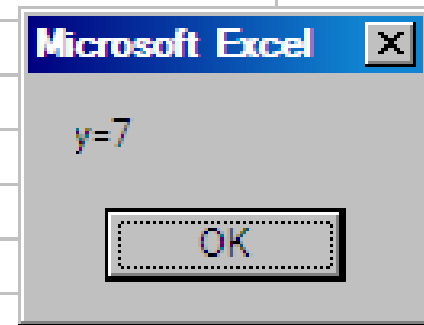
Például könyvtárban Fekete István: VUK c. könyve

- ByRef (Cím) → „Hol van az érték”
  - 4. polc, 3. sor, 17. könyv
- ByVal (Érték) → „Mi az érték”
  - Fekete István: VUK

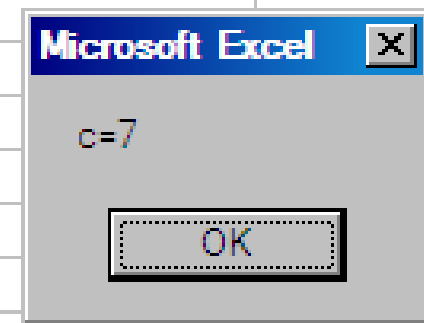
# ByRef

- VBA-ban a ByRef az alapértelmezett (ha nem adjuk meg egyiket sem, úgy veszi, mintha ByRef került volna megadásra)

```
Private Sub UserForm_Initialize()  
    Dim x1 As Byte, x2 As Byte, y As Byte  
    y = 0  
    x1 = 5  
    x2 = 7  
    Call Nagyobb(x1, x2, y)  
    MsgBox ("y=" & y)  
End Sub
```



```
Sub Nagyobb(a As Byte, b As Byte, ByRef c As Byte)  
    If a > b Then  
        c = a  
    Else  
        c = b  
    End If  
    MsgBox ("c=" & c)  
End Sub
```

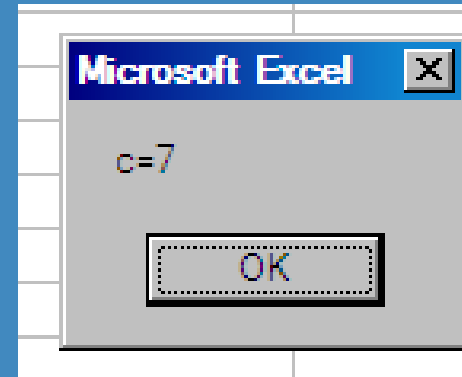
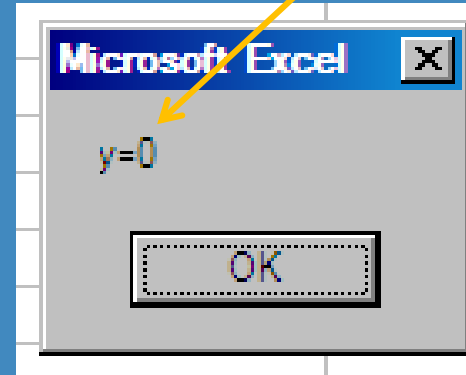


# ByVal

- ByVal: értékszerinti paraméter átadás (Value)
- ByRef: cím szerinti paraméterátadás (Reference)

```
Private Sub UserForm_Initialize()  
    Dim x1 As Byte, x2 As Byte, y As Byte  
    y = 0  
    x1 = 5  
    x2 = 7  
    Call Nagyobb(x1, x2, y)  
    MsgBox ("y=" & y)  
End Sub
```

```
Sub Nagyobb(a As Byte, b As Byte, ByVal c As Byte)  
    If a > b Then  
        c = a  
    Else  
        c = b  
    End If  
    MsgBox ("c=" & c)  
End Sub
```



# Lássuk miért is történt...

- Rajz a táblára...

# Public - Private

- Az alprogramok (Sub, Function) kétféle láthatósággal rendelkeznek:
  - **Public**: bárhonnán látható, és így hívható
  - **Private**: csak azok az alprogramok látják, és tudják hívni, amelyek ugyanazon a Form-on vannak.

A következő slide-on egy olyan VBA program van, amely két Form-ot tartalmaz: frmMain, és frmSajat

# Public - Private

The image shows two VBA code windows side-by-side. The left window is titled 'Munkafüzet1 - frmMain (UserForm)' and contains the following code:

```
Private Sub CommandButton1_Click()  
    Range("A7").Select  
    Selection.Font.Color = vbRed  
End Sub  
  
Private Sub UserForm_Initialize()  
    Dim x1 As Byte, x2 As Byte, y As Byte  
    y = 0  
    x1 = 5  
    x2 = 7  
    Call Nagyobb(x1, x2, y)  
    MsgBox ("y=" & y)  
End Sub  
  
Sub Nagyobb(a As Byte, b As Byte, ByVal c As Byte)  
    If a > b Then  
        c = a  
    Else  
        c = b  
    End If  
    MsgBox ("c=" & c)  
End Sub
```

The right window is titled 'Munkafüzet1 - frmSajat (UserForm)' and contains the following code:

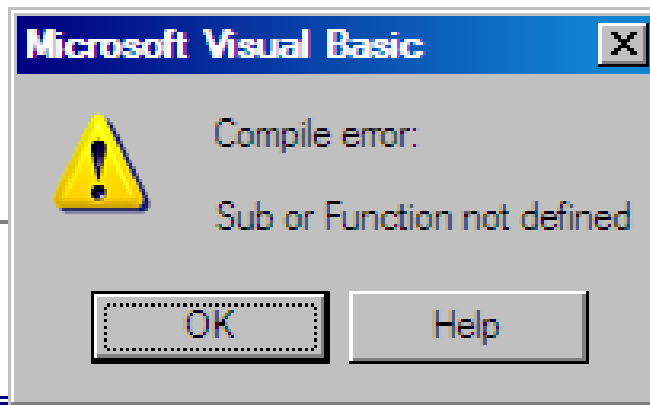
```
Private Sub CommandButton1_Click()  
    KiIr (a)  
    a = "alma"  
    KiIr (a)  
    a = 123.234  
    KiIr (a)  
  
End Sub  
  
Private Sub TextBox1_Change()  
    Load frmSajat  
    frmSajat.Show  
    MsgBox ("most mi van?" )  
    frmSajat.Hide  
    Unload frmSajat  
End Sub
```

A yellow arrow points from the 'Nagyobb' sub in the left window to the 'Private Sub TextBox1\_Change()' in the right window. A large yellow question mark is in the background.

# Private

```
Private Sub UserForm_Initialize()  
    Dim x1 As Byte, x2 As Byte, y As Byte  
    y = 0  
    x1 = 5  
    x2 = 7  
    Call Nagyobb(x1, x2, y)  
    MsgBox ("y=" & y)  
End Sub
```

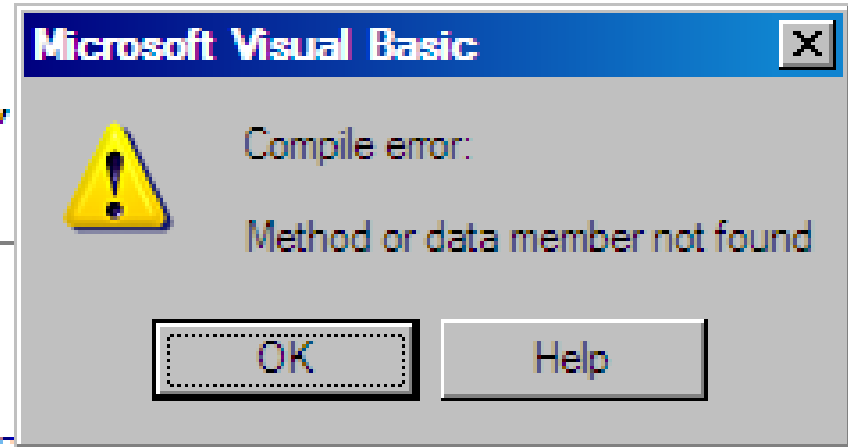
```
Function Vmi(x As Integer) As Boolean  
    a = 123
```



- Ezen a form-on nincs ilyen!

# Private

```
Private Sub UserForm_Initialize()  
    Dim x1 As Byte, x2 As Byte, y As Byte  
    y = 0  
    x1 = 5  
    x2 = 7  
    Call frmSajat.Nagyobb(x1, x2,  
        MsgBox ("y=" & y)  
End Sub
```



```
Function Vmi(x As Integer) As Boolean  
    If x = 123  
        Return True  
    Else  
        Return False  
    End If  
End Function
```

- Így már van, de  
nem találja

```
Private Sub Nagyobb(a As Integer, b As Integer, c As Integer)  
    If a > b Then  
        c = a  
    Else  
        c = b  
    End If  
    MsgBox ("c=" & c)  
End Sub
```



# Public

```
Private Sub UserForm_Initialize()  
    Dim x1 As Byte, x2 As Byte, y As Byte  
    y = 0  
    x1 = 5  
    x2 = 7  
    Call frmSajat.Nagyobb(x1, x2, y)  
    MsgBox ("y=" & y)  
End Sub
```

```
Function Vmi(x As Integer) As Boolean  
    a = 123  
    KiIr (a)  
    a = "alma"  
    KiIr (a)  
    a = 123.234
```

End Sub

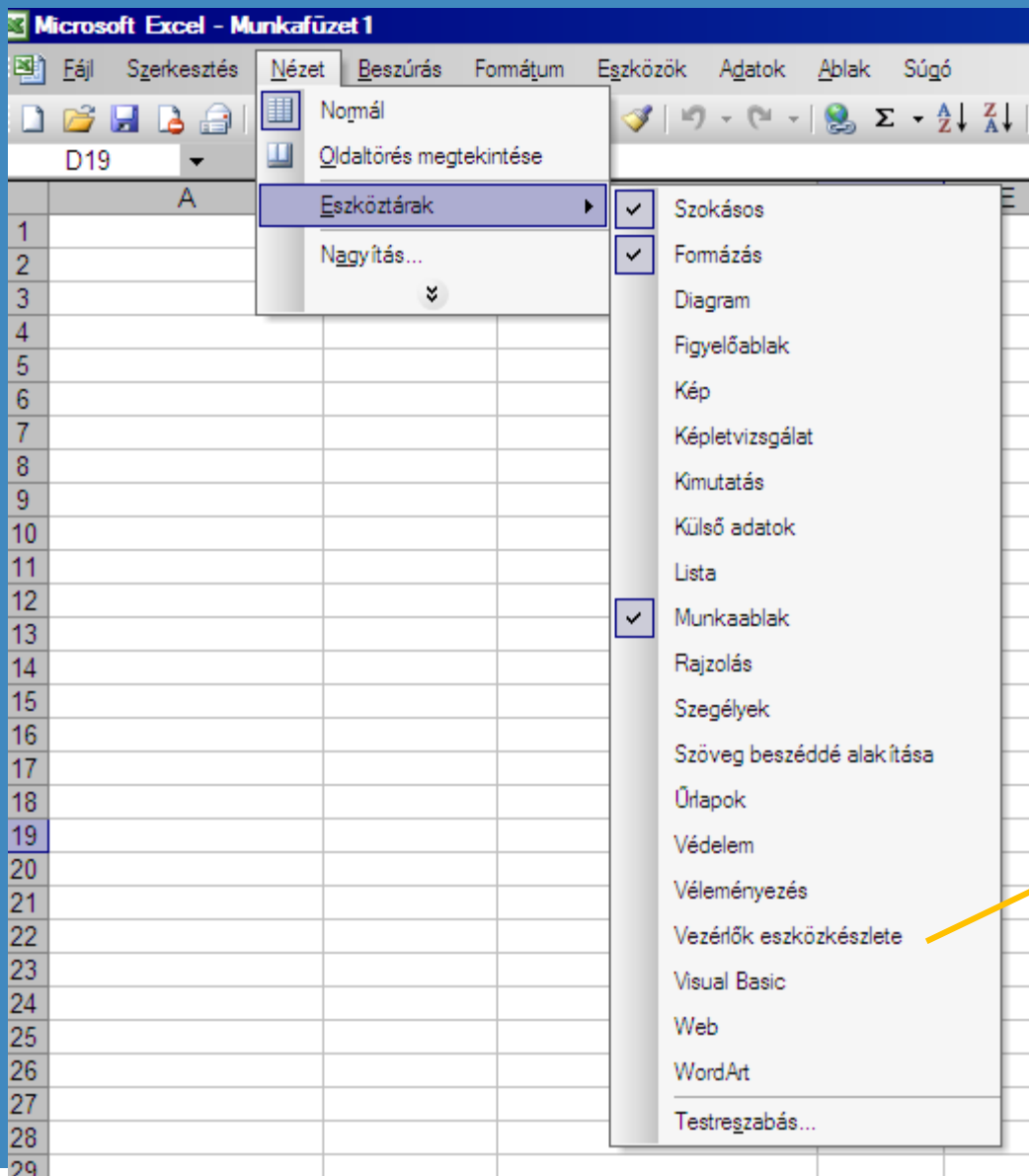
```
Private Sub TextBox1_Change()  
    Load frmSajat  
    frmSajat.Show  
    MsgBox ("most mi van?" )  
    frmSajat.Hide  
    Unload frmSajat
```

End Sub

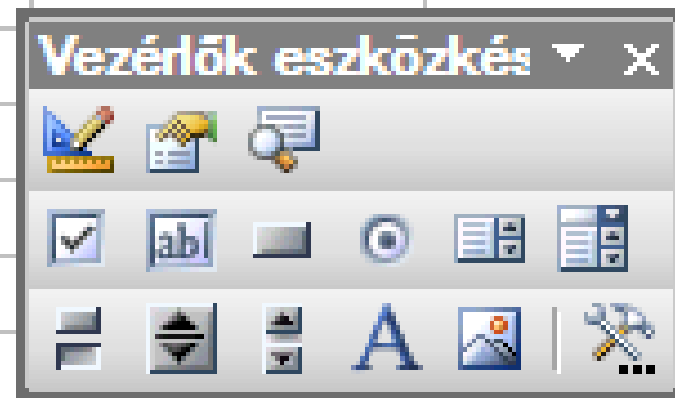
```
Public Sub Nagyobb(a As Byte, b As Byte, ByVal c As Byte)  
    If a > b Then  
        c = a  
    Else  
        c = b  
    End If  
    MsgBox ("c=" & c)  
End Sub
```

Public érvényességi körrel másik form-on levő eljárás,  
vagy függvény is hívható lesz

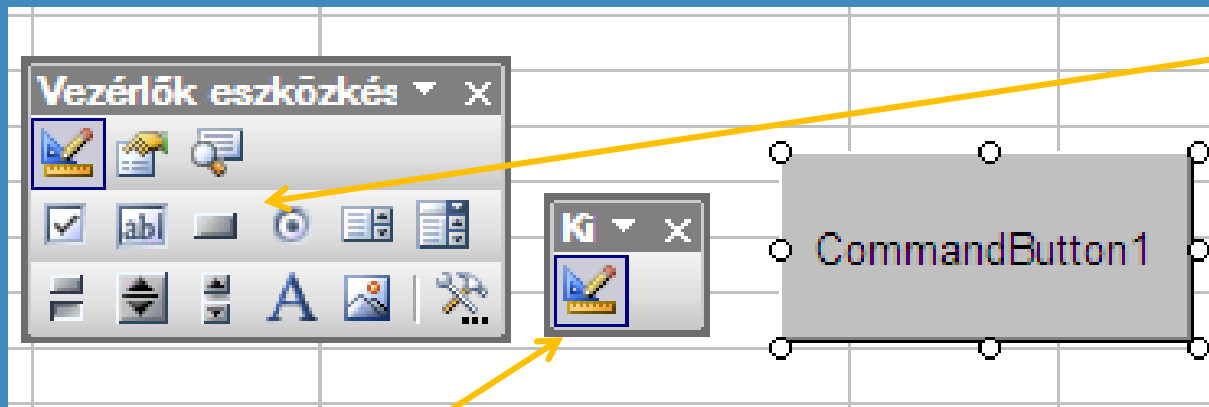
F5-re, F8-ra az indul el, amelyik aktív!



- A munkalap legyen elől
- Nézet  $\rightarrow$  Eszköztárak  $\rightarrow$  Vezérlők eszközkészlete



# Vezérlők eszközkészlete

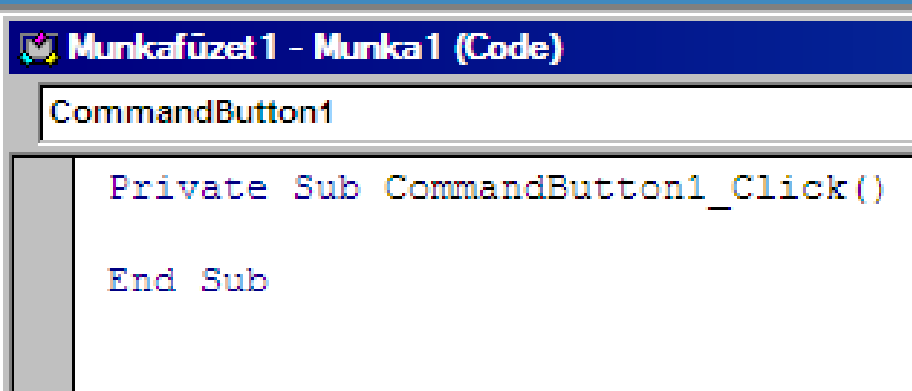


Parancsgomb

Megrajzolni és kicsit várni kell míg megjelenik

Kilépés a tervezésből

- Kettős kattintás a CommandButton1-en
- Meghívni a vezérlőhöz írt eljárást:



VBA Programozás

# A vezérlő eseményének meghívása

Munkafüzet1 - frmMain (Code)

(General)

```
Private Sub cmdKeres_Click()  
    Dim i As Byte  
    Dim k As Byte  
    For i = 1 To 30  
        Cells(i, 1) = CByte(Rnd() * 100 + 20)  
    Next i  
    k = InputBox("Adja meg, mit keres")  
    For i = 1 To 30  
        If Cells(i, 1) = k Then  
            MsgBox ("A keresett adat indexe: " & i)  
            Exit For  
        End If  
    Next i  
End Sub
```

frmMain

Call frmMain.cmdKeres

Munkafüzet1 - Munka1 (Code)

CommandButton1

```
Private Sub CommandButton1_Click()  
    Call frmMain.cmdKeres  
End Sub
```

cmdKeres  
CommandButton1  
Controls  
Copy  
Cut  
Cycle  
DrawBuffer

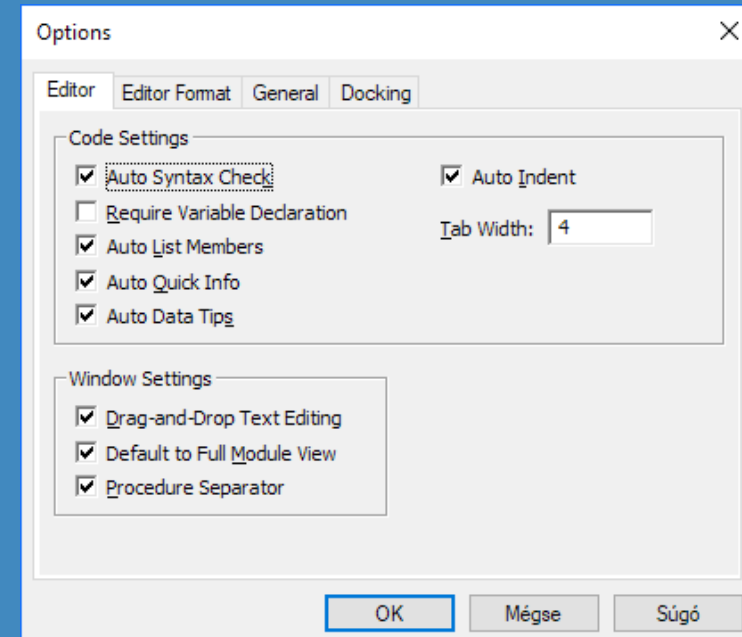
# Hibakezelés VBA-ban

- Háromféle hiba léphet fel:
  - Szintaktikai (Syntax)
  - Fordítási (Compilation)
  - Futási idejű (Runtime)

# Syntax error

- Egy kódsor begépelés közben az Enter megnyomásakor jelentkezhethet:
  - If a > b (nincs Then)
  - For i 2 To 7 (nincs = jel)
  - B = left(„ABCD”, 1 (nincs záró zárójel)
- A szintaktikai hiba mindig 1 sorra korlátozódik!
- Kikapcsolható:

Tools -> Options ->  
Auto Syntax Check



# Compilation error

- Több mint 1 sor után jelentkező hiba
  - If utasítás End If nélkül
  - For utasítás Nexr nélkül
  - Select utasítás End Select nélkül
  - Sub vagy Function hívása rossz paraméterekkel
  - Sub vagy Function ugyanazt a nevet kapja, mint a modul
  - Option Explicit ellenére egy változó nincs deklarálva

# Compilation error

- Védekezés: futtatás előtt le lehet fordítani a programot. Valójában nem fordítás, csak hibaellenőrzés
- Debug -> Compile VBA Project
- Fontos:
  - Az első megtalált hibánál leáll. Azt javítva újra kell indítani a fordítást, amely megint megáll az első megtalált hibánál. HA ki van szűrítve, akkor megvolt a fordítás, és nem talált hibát.
  - A szintaktikai hibákat is megtalálja



# Runtime error

- A program futása közben jelentkezik.
- Például
  - nem létező file,
  - Felhasználó által megadott érvénytelen adat
  - Egy cellában szöveg van szám helyett, ...

# Várt illetve váratlan hiba

(expected ill. unexpected)

- Várt hiba: bekérjük a felhasználótól egy megnyitandó adatfile nevét. Mivel lehetséges, hogy elgépel, ezért itt egy várható, hogy hiba fog fellépni.
- Váratlan hiba: ha programkóddal nem tudjuk kivédeni a hibát, akkor az váratlan hibaként jelentkezik (a VBA hibakezelője fog indulni)

# A VBA által nem detektálható hibák

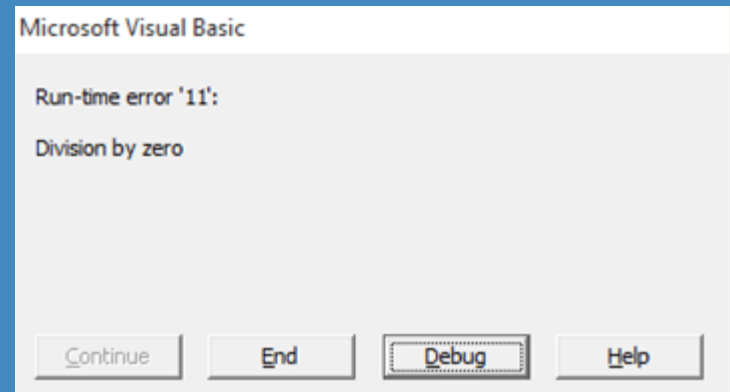
- Egyszerű elgépelések, amelyek azonban szintaktikailag helyesek
  - $x = a + b$  (helyett)
  - $x = a * b$

# On Error utasítás

- Várt hibakezelési célokra használható
- 4 változata van:
  - On Error Goto 0
  - On Error Resume Next
  - On Error Goto [címke]
  - On Error Goto 1

# On Error Goto 0

- Ez az alapértelmezett állapot
- A kód futása megáll annál a sornál, ahol a hiba jelentkezett, és megjelenik egy hibaüzenet
- A felhasználónak a megadott lehetőségek közül választania kell
- Például 0-val való osztás
  - Az üzenet ablak:



# On Error Resume Next

- A VBA figyelmen kívül hagyja a hibát, és a következő utasítással folytatja
- Egyes esetekben igen hasznos, más esetekben inkább értelmetlen, hamis eredményre vezethet
  - Például a 0-való osztás után tovább futva a felhasználó nem tud a hibás adatmegadásról, a program pedig számol tovább

# On Error Goto [címke]

- Jellemzően ez a korrekt megoldása egy hiba lekezelésének

```
Sub UsingGotoLine()  
    On Error Goto eh  
    Dim x As Long, y As Long  
    x = 6  
    y = 6 / 0  
    x = 7  
Done:  
    Exit Sub  
eh:  
    MsgBox "The following error occurred: " & Err.Description  
End Sub
```

# On Error Goto -1

- Nem igazi hibakezelő
- Célja: ennek segítségével törölhető az éppen fellépett hiba
- Az éppen fellépett hiba egy szubrutinból való kilépéskor szintén törlődik



# Az Err objektum

- Ennek segítségével hibához tartozó információkat lehet lekérni. Például:
  - Err.Description: a hiba szövege
  - Err.Number: a hiba száma
  - Err.Source: a projekt neve, ahol a hiba jelentkezett. Sok haszna nincs
  - Err.Raise: függvény, amellyel saját hiba generálható
  - Err.Clear: törli a fellépett hiba számát, szövegét

# Az Error függvény

- Egy meghadott hibaszámú hiba szövegét lehet kinyomtatni
- `Debug.Print Error(11)` „Division by zero”
- `Debug.Print Error(13)` „Type mismatch”

# Hibakezelés során fellépő hiba

```
Sub TwoErrors()  
  
    On Error Goto eh  
  
    ' generate "Type mismatch" error  
    Error (13)  
  
Done:  
    Exit Sub  
eh:  
    On Error Goto eh_other  
    ' generate "Application-defined" error  
    Error (1034)  
Exit Sub  
eh_other:  
    Debug.Print "ehother " & Err.Description  
End Sub
```