Juhász Tibor – Kiss Zsolt:

Programozási ismeretek

(Műszaki Könyvkiadó, 2011, MK–4462-3)

Visual Basic 2008/2010 Express Edition Programozási összefoglaló

Kiegészítések a tankönyvhöz

Bevezetés

A Programozási összefoglaló a Programozási ismeretek tankönyv (Műszaki Könyvkiadó, 2011) kiegészítése. A tankönyvben lehetőség szerint a programozási nyelvektől függetlenítve ismertettük a programozási tudnivalókat. Az alábbiakban bemutatjuk a Visual Basic 2008/2010 nyelvi elemeit és a .NET eszközeit. Megjegyezzük, hogy jelenleg (2011-ben) mind az informatika érettségin, mind a programozás versenyeken (OKTV, Nemes Tihamér) a 2008-as változatot lehet használni.

Az összefoglaló csak a tankönyvhöz kapcsolódó elemekre vonatkozik. A tankönyv anyagát fedi le, de a definíciókban, szintaxisban az általánosság kedvéért megemlítünk olyan fogalmakat is, amelyek nem szerepelnek a könyvben. Az azonos célra használható nyelvi elemek közül előnyben részesítettük azokat, amelyek illeszkednek az objektumorientált szemléletmódhoz (például a véletlenszám-generálásnál). Az egyes objektumok, objektumosztályok tulajdonságai és metódusai közül csak a legfontosabbakat ismertetjük. Sok esetben nem adjuk meg a tulajdonságok lehetséges értékeit, az intelligens súgó kilistázza a választható felsoroláselemeket. A részletesebb ismertetést a Visual Basic súgójában, illetve a programozási nyelv dokumentációjában találjuk.

A Súgó tartalomjegyzékéből a következő bejegyzésekre hívjuk fel a figyelmet:

A programozási nyelv leírása: Visual Basic/Reference/Visual Basic Reference

Az objektumosztályok ismertetése: .NET Framework/Class Library

A Microsoft Development Network webhelyén (http://msdn.microsoft.com/en-us/library) a tartalomjegyzék következő bejegyzéseinél olvashatjuk a programozási nyelv és az objektumosztályok ismertetését:

MSDN Library/Development Tools and Languages/Visual Studio 2008/Visual Studio/Visual Basic/Reference

MSDN Library/NET Development/.NET Framework 3.5/.NET Framework Class Library

Hasonló módon találjuk meg a Visual Basic 2010 és a .Net Framework 4-es verziójának leírását.

A Visual Basic 2008/2010 dokumentációját a The Microsoft Visual Basic Language Specification Version 9.0 tartalmazza.

Végezetül megemlítjük, hogy rengeteg példaprogram található a VB Helper webhelyén (http://www.vb-helper.com/index_categories.html), illetve számos további webhelyen.

Jelölések

A forráskódú részleteket Courier betűtípus jelöli az összefoglalóban.
A szintaxisban szereplő dőlt betűs részeket a megfelelő tartalommal kell helyettesíteni.
A három pont (...) arra utal, hogy az előtte lévő rész értelemszerűen tetszőlegesen sokszor ismétlődhet.
A szögletes zárójelben lévő részeket nem kötelező beírni a forráskódba. A szögletes zárójel nem része a szintaxisnak.
A kapcsos zárójelben lévő, függőleges vonallal elválasztott opciók közül az egyiket kötelező alkalmazni. A kapcsos zárójel és a függőleges vonal nem része a szintaxisnak.

Névterek a .NET-ben

A .NET több ezerre tehető osztálykönyvtára hierarchikus rendszert alkot. A rendszer egy-egy csomópontját névtérnek nevezzük. Egy névtér tartalmazhat további névtereket, illetve osztálydefiníciókat vagy struktúrákat. A névterek rendszerét egy háttértár mappaszerkezetéhez hasonlóan képzelhetjük el. A névterek rendszere lehetővé teszi az osztályok csoportosítását, megkönnyíti az áttekintést és megakadályozza az azonosítók ütközését.
A .NET-ben egy névtér azonosítóinak a használatához importálnunk kell a névteret a projektbe (Imports utasítás), vagy teljesen minősített hivatkozást kell alkalmaznunk, például: My.Computer.FileSystem.CurrentDirectory. Egy importált névtér alá tartozó névterek esetén nem szükséges a teljesen minősített hivatkozás kiírása, például: Threading.Timer (a System.Threading.Timer helyett).
Új projekt létrehozása esetén a fejlesztőrendszer alapértelmezés szerint automatikusan importálja a következő névtereket:

- Microsoft.VisualBasic
- System
- System.Collections
- System.Collections.Generic
- System.Data
- System.Diagnostic
- System.Linq
- System.Xml.Linq

Windows alkalmazás létrehozásánál még a következő névterek is importálásra kerülnek:

- System.Drawing
- System.Windows.Forms

Ezeknek a névtereknek az elemeinél elhagyható a minősítés, illetve nincs szükség az Imports utasításra.

Elnevezési konvenciók

A tankönyvben nem tárgyaljuk a Namespace utasítást, így egy projekt egyetlen névtérnek felel meg. Nem térünk ki az assembly-k ismertetésére sem. Tárgyalásmódunkban egyetlen projektet egyetlen assembly valósít meg. Összefoglalónkban ezért **névtér és assembly helyett is projektet írunk** (például projektszintű hozzáférés).
Az objektumosztályok megosztott (shared) tulajdonságait **osztálytulajdonságnak**, sztatikus (shared) metódusait pedig **osztálymetódusnak** nevezzük. Az osztálymetódusokat és osztálytulajdonságokat az objektumosztály nevével kell minősíteni.
A struktúrák megosztott (shared) mezőit **megosztott tulajdonságnak** (mezőnek), közös (shared) metódusait pedig **sztatikus metódusnak** nevezzük. A megosztott tulajdonságokat és sztatikus metódusokat a struktúra nevével kell minősíteni.

A tankönyvben szereplő hivatkozások a programozási összefoglaló egyes részeire:

Alapismeretek

A nyelv szintaxisa

A Visual Basic forráskódjában általában minden utasítást külön sorba írunk.
Az utasítás folytatása a következő sorban: szóköz aláhúzásjel Enter

```
utasítás _
     az utasítás folytatása
```

Az aláhúzásjelet semmi nem követheti a sorban!
Szükség esetén egy sorba több utasítás írható, kettősponttal elválasztva:

```
utasítás1 : utasítás2 : …
```

A forráskódban aposztrófjel (') után megjegyzés következhet:

```
[utasítás ]' megjegyzés
```

A Standard eszköztár Comment/Uncomment gombjai () a kijelölt sorokból megjegyzéseket készítenek, illetve törlik a megjegyzésjelet (aposztrófjelet).

Kulcsszavak

A Visual Basic a kulcsszavakban nem különbözteti meg a kisbetűket a nagybetűktől.

Foglalt kulcsszavak

AddHandler	AddressOf	Alias	And	AndAlso	As	Boolean	ByRef
Byte	ByVal	Call	Case	Catch	CBool	CByte	CChar
CDate	CDbl	CDec	Char	CInt	Class	CLng	CObj
Const	Continue	CSByte	CShort	CSng	CStr	CType	CUInt
CULng	CUShort	Date	Decimal	Declare	Default	Delegate	Dim
DirectCast	Do	Double	Each	Else	ElseIf	End	EndIf
Enum	Erase	Error	Event	Exit	False	Finally	For
Friend	Function	Get	GetType	GetXMLNamespace	Global	GoSub	GoTo
Handles	If	Implements	Imports	In	Inherits	Integer	Interface
Is	IsNot	Let	Lib	Like	Long	Loop	Me
Mod	Module	MustInherit	MustOverride	MyBase	MyClass	Namespace	Narrowing
New	Next	Not	Nothing	NotInheritable	NotOverridable	Object	Of
On	Operator	Option	Optional	Or	OrElse	Out	Overloads
Overridable	Overrides	ParamArray	Partial	Private	Property	Protected	Public
RaiseEvent	ReadOnly	ReDim	Rem	RemoveHandler	Resume	Return	SByte
Select	Set	Shadows	Shared	Short	Single	Static	Step
Stop	String	Structure	Sub	SyncLock	Then	Throw	To
True	Try	TryCast	TypeOf	UInteger	ULong	UShort	Using
Variant	Wend	When	While	Widening	With	WithEvents	WriteOnly
Xor	#Const	#Else	#ElseIf	#End	#If	=	&
&=	*	*=	/	/=	\	\=	^
^=	+	+=	-	-=	>>, >>=	<<, <<=	

Nem foglalt kulcsszavak

– de nem célszerű azonosítóként alkalmazni a következőket:

Aggregate	Ansi	Assembly	Auto	Binary	Compare	Custom	Distinct
Equals	Explicit	From	Group By	Group Join	Into	IsFalse	IsTrue
Join	Key	Mid	Off	Order By	Preserve	Skip	Skip While
Strict	Take	Take While	Text	Unicode	Until	Where	#ExternalSource
#Region							

A programok szerkezete

A Visual Basic programok forráskódja kódfájlokban helyezkedik el. Egy kódfájl programmodulokat tartalmaz. A programmodulokat röviden moduloknak nevezzük.
A Visual Basic moduljai:

– osztályok,
– struktúrák,
– standard modulok (további változókkal, eljárásokkal, függvényekkel),
– interfészek (ezeket a tankönyvben nem tárgyaljuk).

Egy kódfájlban több modult is definiálhatunk. A program végrehajtható utasításai csak modulokban helyezkedhetnek el!

A kódfájl szerkezete

		Megjegyzés

Fájlszintű elemek	`[`*`Option utasítások`*`]`
	`[`*`Imports utasítások`*`]`
	`projektszintű elemek`
		A sorrend kötött!
		A forráskód további, a fordítóprogramnak szóló üzeneteket, úgynevezett direktívákat is tartalmazhat.

Projektszintű elemek, köztük a programmodulok (osztály, struktúra, standard modul)	`[Class … End Class]`
	`[Structure … End Structure]`
	`[Module … End Module]`
	`[Enum … End Enum]`
		Osztályok, struktúrák, standard modulok, felsorolások.
		A standard modulok és a felsorolások nem ágyazhatók egymásba!
		Egy kódfájlban több projektszintű elemet is definiálhatunk, tetszőleges sorrendben.

Modulszintű elemek (a programmodulokban helyezkednek el)	`[Enum … End Enum]`
	`[Structure … End Structure]`
	`[`*`konstansok deklarációi`*`]`
	`[`*`változók deklarációi`*`]`
	`[Function … End Function]`
	`[Sub … End Sub]`
		Felsorolások, struktúrák, konstansok, változók deklarációi, függvények, eljárások definíciói.
		A függvények, eljárások, felsorolások nem ágyazhatók egymásba!

Alprogram-szintű elemek (alprogramokban helyezkednek el)	`[`*`lokális konstansok deklarációi`*`]`
	`[`*`változók deklarációi`*`]`
	`[`*`végrehajtható utasítások`*`]`
		Minden végrehajtható utasításnak alprogramban (függvény, eljárás) kell elhelyezkednie!

Option utasítások

Az Option utasítások a fordítóprogram számára szóló üzenetek (direktívák). Értéküket beállíthatjuk a projekt tulajdonságlapján vagy a forráskódban.

`Option Explicit {On | Off}`	On: kötelező a változók deklarálása.
`Option Strict {On | Off}`	On: kötelező a változó típusának megadása, szűkebb típusra nem végzi el az automatikus típuskonverziót!
`Option Compare {Binary | Text}`	Binary: sztringek összehasonlítása a Unicode alapján. Text: sztringek összehasonlítása a területi beállítások alapján.
`Option Infer {On | Off}`	On: nem kötelező a típust megadni a deklarációban, a változó értékadása alapján állapítja meg.

Megjegyzés: a menüben (Tools/Options/Projects and Solutions/VB Defaults) módosítva csak a módosítás után létrehozott projektekre vonatkoznak!

Imports utasítások

`Imports` *`névtér`*`[.`*`osztály`*`]`	Importálja a névtérhez tartozó azonosítókat a programba, így az azonosítókra minősítés nélkül hivatkozhatunk.

Azonosítók

Az azonosítók elnevezése

A konstansok, változók, eljárások, függvények, objektumok, osztályok stb. azonosítói

- Unicode-betűt, számjegyet vagy aláhúzásjelet tartalmazhatnak;
- betűvel vagy aláhúzásjellel (_) kezdődhetnek;
- legfeljebb 1023 karakterből állhatnak;
- nem egyezhetnek meg valamely foglalt kulcsszóval.

A Visual Basic az azonosítókban nem különbözteti meg a kisbetűket a nagybetűktől. Az azonosítók ékezetes karaktereket is tartalmazhatnak.
Az eseménykezelő eljárások paraméterei automatikusan a „sender” és az „e” azonosítót kapják. Átnevezésük nélkül nem használhatjuk ugyanezeket az azonosítókat a lokális deklarációkban.

Az azonosítók hatóköre[1]

Blokkszintű hatókör: a deklarációtól az utasításblokk végéig terjed.
Utasításblokkok: Do … Loop, For [Each] … Next, If … End If, Select … End Select, Try … End Try, With … End With
A blokkon belül deklarált változók blokkszintű hatókörrel rendelkeznek.
Egy blokkban deklarált változó azonosítója nem egyezhet meg a blokkot tartalmazó alprogram lokális változóinak azonosítójával!
Eljárásszintű hatókör: a deklarációtól az alprogram (eljárás/függvény) végéig terjed.
Az alprogramon belül deklarált változók eljárásszintű hatókörrel rendelkeznek (lokális változók).
Az alprogramban deklarált változók csak Private hozzáférésűek lehetnek (a Public nem használható!).

[1] Scope

Modulszintű hatókör: a teljes programmodulra kiterjed (a deklarálás helyétől függetlenül)
Modul (programmodul): osztály, struktúra, standard modul.
A programmodulon belül, de az alprogramokon kívül deklarált változók alapértelmezés szerint modulszintű hatókörrel rendelkeznek (tagváltozók).
A programmodulban definiált alprogramok alapértelmezés szerint modulszintű hatókörrel rendelkeznek.
A modulszintű hatókörrel rendelkező azonosítókra minősítés nélkül hivatkozhatunk a programmodul eljárásaiban.
Egy objektumosztály definíciója esetén a modulszintű hatókört szokás osztályszintű hatókörnek nevezni (ne keverjük össze az osztály megosztott, Shared változóival).
Megjegyzés: modulszintű hatókör esetén olyan azonosítóra is hivatkozhatunk, amelynek a deklarálása/definíciója csak később következik a forráskódban.
Projektszintű hatókör: a projekt minden moduljára kiterjed.
A Public (vagy Friend) hozzáférésű, modulszinten deklarált változók projektszintű hatókörrel rendelkeznek (globális változók).
A projektszintű hatókörrel rendelkező változókra a deklaráció helyétől eltérő projektben is hivatkozhatunk (lásd alább, a Hozzáférési módoknál).

Az azonosítók láthatósága

Azonos nevek esetén, ha átfedik egymást a hatókörök, akkor minősítés nélkül a szűkebb hatókörű azonosítóra vonatkozik a hivatkozás. A tágabb hatókörű azonosítót a programmodul (illetve a projekt és a programmodul) nevével minősítve érjük el.
Az osztály vagy struktúra eljárásaiban ütközés esetén a modulszintű változókra a Me minősítéssel hivatkozunk: Me.*változónév*.

Hozzáférési módok[2]

A hozzáférési mód módosítja az azonosítók hatókörét. **Az alábbiakban hangsúlyozottan csak a tankönyvhöz kapcsolódó ismereteket foglaljuk össze!**
A hozzáférési módot a deklarációban, illetve a programmodulok és alprogramok fejében szabályozhatjuk (például Public Sub …).
A hozzáférési módok meghatározása függ a deklaráció helyétől:
(1) projektszintű elemek: programmodulon kívül (Class … End Class, Structure … End Structure, Module … End Module utasításokon kívül) helyezkednek el;
(2) modulszintű elemek: programmodulon belül, de alprogramon kívül helyezkednek el;
(3) eljárásszintű/blokkszintű elemek: alprogramban, illetve utasításblokkban helyezkednek el.
Private hozzáférési mód: az azonosító csak a deklarációt tartalmazó programmodulon (osztály, standard modul, struktúra) belül érhető el.
A Private hozzáférési mód csak programmodul-szinten alkalmazható (programmodulra vonatkozóan, illetve alprogramon belül nem írható elő).
Private hozzáférésű lehet: programmodulon belül deklarált osztály és struktúra, struktúra mezője, modul/osztályszintű változó, konstans, eljárás, függvény, felsorolás.
Public hozzáférési mód: az azonosító korlátozás nélkül, bárhonnan elérhető (másik projektből a projektre mutató hivatkozás felvételével).
A Public hozzáférési mód csak programmodul-, illetve projektszinten alkalmazható (alprogramon belül nem írható elő).
Public hozzáférésű lehet: standard modul, osztály, modul/osztályszintű változó, konstans, struktúra, struktúra mezője (változója), eljárás, függvény, felsorolás.
Hivatkozás felvétele más projektre: Project/Add Reference, a Projects panelen kiválasztjuk a megfelelő projektet, OK.
Friend hozzáférési mód: az azonosító csak a projekt kódfájljaiból érhető el[3], azaz más projektből nem hivatkozhatunk rá.
A Friend hozzáférési mód csak programmodul-, illetve projektszinten alkalmazható (alprogramon belül nem írható elő).
Friend hozzáférésű lehet: standard modul, osztály, modul/osztályszintű változó, konstans, struktúra, struktúra mezője, eljárás, függvény, felsorolás.

[2] Access levels
[3] Pontosabban csak az assembly-n belül érhető el. Célszerűen és általában egy névtér (projekt) egy assembly-nek felel meg.

Alapértelmezett hozzáférési módok

Programelem	Alapértelmezett hozzáférési mód		
	Projektszinten (1)	**Modulszinten (2)**	**Eljárásszinten/blokkszinten (3)**
	deklarálva		
Konstans	–	Private, struktúrában Public	Public (nem módosítható)
Változó (Dim utasítással deklarálva)			
Alprogram (eljárás, függvény)	–	Public	–
Alprogram paramétere	–	–	Public (nem módosítható)
Standard modul (Module)	Friend	–	–
Osztály	Friend	Public	–
Struktúra	Friend	Public	–
Felsorolás (Enumeration)	Friend	Public	–

Megjegyzés: egy alprogram paraméterei alapértelmezés szerint Public hozzáférésűek, ezért nem lehetnek a programmodulon kívül deklarált Private vagy Friend elérésűek. Szükség esetén a típust is Public hozzáféréssel kell ellátni, vagy pedig az alprogramnak adjunk Private, illetve Friend minősítést!

Elemi típusok

Elemi típusok és literáljaik

Névtér: System
A sztringek objektumok, a többi elemi típus pedig struktúra!

				Megjegyzés
A legfontosabb elemi típusok	Karakter:	Char	2 bájt	Egyetlen Unicode-karakter
	Sztring:	String	a méret függ az	Legfeljebb 2 milliárd Unicode-karakter (objektum!)
	Logikai:	Boolean	implementációtól	True vagy False
	Dátum-idő:	Date	8 bájt	i.sz. 0001 jan. 1. 0:00:00-tól 9999. dec. 31. 23:59:59-ig
	Egész típusok (kettes komplemens kódban):			
	Bájt	Byte	1 bájt	0-tól 255-ig
	Rövid egész	Short (Int16)	2 bájt	–32768-tól +32767-ig
	Egész:	Integer (Int32)	4 bájt	–2147483648-tól +2147483647-ig
	Hosszú egész:	Long (Int64)	8 bájt	$-9{,}2\cdot10^{18}$-tól $+9{,}2\cdot10^{18}$-ig
	Valós (lebegőpontos) típusok:			
	Egyszeres pontosságú:	Single	4 bájt	$\pm1{,}4\cdot10^{-45}$-től $\pm3{,}4\cdot10^{38}$-ig (7–8 értékes jeggyel)
	Dupla pontosságú:	Double	8 bájt	$\pm4{,}9\cdot10^{-324}$-től $\pm1{,}8\cdot10^{308}$-ig (16–17 értékes jeggyel)
	Fixpontos típus:			
	Decimális	Decimal	16 bájt	$\pm7{,}9\cdot10^{28}$ között (29 értékes jeggyel), főleg pénzügyi számításokhoz
Numerikus értékek literáljai	Előjeles egész vagy tizedestört alakban felírt számok Lebegőpontos alakban felírt számok, például: –25.67E–12			**A forráskódban tizedespontot írunk!**
Hexadecimális érték literálja	`&H`*`xxxx`*			*xxxx*: hexadecimális számjegyek a típusnak megfelelő számban. Szükség esetén alkalmazzunk típusazonosító karaktert! Az előjeles egész típusoknál a hexadecimális érték kettes komplemens kódban kerül értelmezésre!
Karaktersorozat literál	`"`*`karaktersorozat`*`"`			A sztringen belüli idézőjelet két idézőjel helyettesíti.
Dátum/idő literál	`#[`*`hónap/nap/év`*`][ `*`óra`*`:`*`perc`*`[:`*`másodperc`*`]]#`			Vagy a dátumot vagy az időt kötelező megadni.
Típusazonosító karakterek	Short: S Integer: I Long: L Karakter: C		Single: F Double: R Decimal: D	Közvetlenül a literál mögé írjuk (szóköz nélkül!).

Megjegyzés: a False kódja 0, a True kódja –1. A CBool függvény azonban minden nem 0 numerikus értéket True-ra konvertál.

A numerikus típusok (struktúrák) tulajdonságai és metódusai

A ToString metódus kivételével megosztott tulajdonságok, illetve sztatikus metódusok.
Hívás: *típus*.*tulajdonságnév*, *típus*.*metódusnév*(*argumentumok*)

típus.MaxValue, *típus*.MinValue	A típus legkisebb, illetve legnagyobb értéke.
típus.Epsilon	Valós típusok tárolható legkisebb pozitív értéke.
típus.PositiveInfinity, *típus*.NegativeInfinity *típus*.IsInfinity(*kifejezés*), *típus*.IsPositiveInfinity(*kifejezés*), *típus*.IsNegativeInfinity(*kifejezés*)	A valós MaxValue, illetve MinValue értékét túllépő műveletek eredménye, illetve vizsgálata.
típus.NaN, *típus*.IsNaN(*kifejezés*)	A valós típusú 0/0 eredménye, illetve vizsgálata.
ToString(["*formátumkód*"])	Sztringgé alakítja a numerikus értéket. A *formátumkód* értelmezését lásd a sztringeknél.

A Char típus (struktúra) sztatikus metódusai

Hivatkozás: Char.*metódusnév*(*karakter*)
c: karakter

IsControl(*c*), IsDigit(*c*), IsLetter(*c*), IsLetterOrDigit(*c*), IsLower(*c*), IsPunctuation(*c*), IsSeparator(*c*), IsUpper(*c*), IsWhiteSpace(c*)*	Kontrol-karakter, számjegy, betű, betű vagy számjegy, kisbetű, írásjel, szóköz vagy Enter, nagybetű, szóköz vagy Enter vagy tabulátor vizsgálata (True/False). Egy sztring *ind* indexű karakterére is meghívhatók Char.*metódusnév*(*szrtring*, *ind*) formában.
ToLower(*c*), ToUpper(*c*)	Konverzió kisbetűvé, nagybetűvé.

Dátum és idő

Kezdőértékadáshoz a konstruktor is használható: New DateTime(*év*, *hó*, *nap*[, *óra*, *perc*, *másodperc*[, *ezredmásodperc*]])
Az argumentumok egész típusú kifejezések.

A DateTime struktúra tulajdonságai

Hivatkozás: *változónév*.*tulajdonságnév*, megosztott tulajdonságoknál: DateTime.*tulajdonságnév*

Date, TimeOfDay	A változó értékének dátum, illetve idő része.
Year, Month, Day, Hour, Minute, Second, Millisecond	A változó értékének év, hónap, nap, óra, perc, másodperc, ezredmásodperc része.
DayOfWeek *változónév*.DayOfWeek.ToString()	A hét napja (0: vasárnap, 6: szombat). A hét napjának angol elnevezése.
DayOfYear	Az év hányadik napja.
DateTime.Today, DateTime.Now	Rendszerdátum, rendszeridő (dátum/idő), megosztott tulajdonságok.

A DateTime struktúra metódusai

Hivatkozás: *változónév*.*metódusnév*(*argumentumok*), sztatikus metódusnál: DateTime.*metódusnév*(*argumentumok*)
i: egész típusú érték, *d*: dupla pontosságú érték

AddYears(*i*), AddMonths(*i*), AddDays(*d*), AddHours(*d*), AddMinutes(*d*), AddSeconds(*d*), AddMilliseconds(*d*)	A megadott értéket hozzáadja a változó év, hónap, nap, óra, perc, másodperc, ezredmásodperc részéhez. Az AddYears esetén csak az év változik (például nem veszi figyelembe a szökőévet).
ToString([*formátumsztring*])	Sztringgé alakítja a változó értékét a megadott formátumban. A formátumsztringekre példák találhatók a Dátumformátumok.vb kódfájlban.
DateTime.DaysInMonth(*év*, *hónap*)	A hónap napjainak a száma (sztatikus metódus).
DateTime.IsLeapYear(*év*)	Szökőév-e a megadott érték (True/False, sztatikus metódus).
DateTime.Parse(*sztring*)	Dátum/idővé alakítja a sztringet (sztatikus metódus).
DateTime.TryParse(*sztring*, *változó*)	A változóba beírja a sztring dátum/idővé alakított értékét, ha lehet. Visszatérési értéke (True/False) jelzi az átalakítás sikerét (sztatikus metódus).

A DateTime típussal kapcsolatos függvények

DateAdd(*mértékegység*, *időtartam*, *dátum*)	A *dátum*-hoz hozzáadja a *mértékegység*-ben mért *időtartam*-ot. Mértékegységsztring: "yyyy": év; "m": hónap; "d": nap; "w": hétköznap; "h": óra, "n": perc (!); "s": másodperc; "q": negyedév; "ww": hét. A mértékegységsztring helyett használhatjuk a DateInterval felsorolás konstansait (például: DateInterval.Day). A felsorolás lehetséges értékeit az intelligens súgó megjeleníti a forráskódban.
DateDiff(*mértékegység*, *dátum1*, *dátum2*)	A *dátum2–dátum1* (!) értéke a megadott mértékegységben kifejezve. A mértékegységsztring értelmezését lásd a DateAdd függvénynél.
DateSerial(*év*, *hónap*, *nap*)	A megadott értékekből DateTime típusú értéket képez. A hónap és a nap lehet negatív, illetve 12-nél vagy 31-nél nagyobb is.
DateValue(*sztring*)	DateTime típusúvá alakítja a sztringet.
IsDate(*kifejezés*)	DateTime típusúvá alakítható-e a kifejezés (True/False).
MonthName(*kifejezés*)	Megadja a *kifejezés*-nek megfelelő sorszámú hónap nevét.
TimeSerial(*óra*, *perc*, *másodperc*)	A megadott értékből DateTime típusú értéket képez. A perc és másodperc lehet kisebb, mint nulla, illetve nagyobb, mint 60.
TimeValue(*sztring*)	A sztringet DateTime értékké konvertálja.
WeekDayName(WeekDay(dátum))	Megadja a hét napjának a nevét.

Megjegyzés: DateTime típusú értékekkel közvetlenül is végezhetünk műveleteket, de két dátum különbsége, illetve az összeadásnál a második operandus TimeSpan (időtartam) típusú érték.

A sztring típus (objektumosztály)

Névtér: System
A sztringek objektumok. Karakterobjektumok (!) sorozatát tartalmazzák. Egy sztring hossza legfeljebb 2 milliárd karakter lehet.
A sztring-literált idézőjelek közé zárjuk. A literálon belül az idézőjelet duplázással jelezzük:
```
sztringváltozó = "Ez egy idézet: ""idézet"""
```
Az üres sztringet két idézőjel jelöli: ""
A sztringobjektumot létrehozhatjuk a New operátorral, illetve a kezdőérték megadásával:
```
Dim S1, S2 As String
S1 = New String("abc")
S2 = "def"
Dim S3 As String = "ghi"
Dim S4 As String = New String("jkl")
```
A konstruktor argumentumaként megadhatunk karakterekből álló tömböt is:
```
Dim S5 As String = New String(karaktertömb[, kezdőindex, darab])
```
Ekkor a sztring értékét a karaktertömb *darab* számú eleméből fűzi össze a *kezdőindex*-től kezdve.
A sztring karaktereit 0-tól kezdve indexeljük. A sztring egy karakterére a Chars(*indexkifejezés*) csak olvasható tulajdonsággal hivatkozhatunk:
`változónév.Chars(indexkifejezés)`, vagy röviden: `változónév(indexkifejezés)`
A karakterekre történő hivatkozások nem használhatók a karakterek módosítására! Hibához vezet, ha értékadó utasítás bal oldalán helyezkednek el!
A sztring módosításakor egy új sztringobjektum jön létre a módosított tartalommal. A program átállítja a sztringváltozót az új objektumra, és törli a régit. Ezért a többi objektummal ellentétben az S2 = S1 értékadás után az S2 nem az S1-re mutat, hanem megkapja az S1 által tárolt sztringet.
Megjegyzés: módosítható sztringet a StringBuilder osztály objektumai tárolnak.

A sztringobjektum tulajdonságai és metódusai

Hívatkozás: *sztringváltozónév.tulajdonságnév*, *sztringváltozónév.metódusnév*(*argumentumok*). A metódusok általában függvények, tehát nem az eredeti változót módosítják, hanem a visszatérési értékük lesz az új sztring!

A *hasonlítás* paraméter fontosabb értékei (az intelligens súgó megjeleníti a forráskódban):

CurrentCulture	a területi beállításoknál megadott nyelv ábécéjének megfelelő összehasonlítás, megkülönbözteti a kis- és nagybetűket.
CurrentCultureIgnoreCase	a területi beállításoknál megadott nyelv ábécéjének megfelelő összehasonlítás, egyezőnek tekinti a kis- és nagybetűket.
Ordinal	a karakterkódnak megfelelő összehasonlítás, megkülönbözteti a kis- és nagybetűket.
OrdinalIgnoreCase	a karakterkódnak megfelelő összehasonlítás, egyezőnek tekinti a kis- és nagybetűket.

Length	A sztring hossza.
Contains(*sztring*)	Tartalmazza-e a változó a *sztring*-et (True/False).
CopyTo(*kezdőindex*, *karaktertömb*, *célindex*, *darab*)	A *kezdőindex*-től átmásol *darab* karaktert a *karaktertömbbe* a *célindex*-től kezdve.
EndsWith(*sztring*[, *hasonlítás*])	A változó a *sztring* sztringgel végződik-e (True/False).
IndexOf(*sztring*[, *kezdőindex*[, *elemszám*[, *hasonlítás*]]])	Megadja a *sztring* sztring első előfordulásának pozícióját a *kezdőindex*-től kezdve (*elemszám* darab karakteren keresztül keres).

IndexOfAny(*tömb*[, *kezdőindex*[, *elemszám*]])	Megadja a *tömb* karaktertömb bármely elemének első előfordulását a *kezdőindex*-től kezdve (*elemszám* darab karakteren keresztül keres). Megkülönbözteti a kis- és nagybetűket egymástól.
Insert(*kezdőindex*, *sztring*)	A sztring-et beilleszti a kezdőindex-től kezdve.
LastIndexOf, LastIndexOfAny	Ugyanaz, mint az IndexOf, illetve IndexOfAny, csak a sztring végéről kezdi a keresést.
PadLeft(*teljeshossz*[, *karakter*])	*Teljeshossz* szélességen jobbra zárja a változó karaktereit. A megmaradó üres helyeket *karakter* karakterrel tölti fel (alapértelmezett: szóköz).
PadRight	Ugyanaz mint a PadLeft, csak balra zár.
Remove(*kezdőidex*[, *darab*])	A *kezdőindex*-től kezdve *darab* karaktert töröl (alapértelmezett: a végéig töröl).
Replace(*sztring1*, *sztring2*)	A *sztring1* összes előfordulását lecseréli *sztring2*-re.
StartsWith	Ugyanaz, mint az EndWith, csak a sztring elejét vizsgálja meg.
Substring(*kezdőindex*[, *darab*])	A *kezdőindex*-től kezdve kiemel *darab* karakterből álló rész-sztringet (alapértelmezett: a sztring végéig).
ToCharArray([*kezdőindex*, *darab*])	A *kezdőindex*-től kezdve *darab* számú karaktert helyez el egy karaktertömbben (alapértelmezett: az egész sztring).
ToLower(), ToUpper()	Kisbetűssé, illetve nagybetűssé alakítja a sztringet.
Trim([*karaktertömb*])	Eltávolítja a karaktertömb-ben szereplő karaktereket a sztring elejéről és végéről (alapértelmezett: szóköz, Enter, tabulátor).
TrimStart, TrimEnd	Ugyanaz mint a Trim, de csak az elejéről, illetve a végéről hagyja el.

A következő tömbmetódusok (lásd ott) értelemszerűen alkalmazhatók a sztringekre is: All, Any, Count, Distinct, Except, Intersect, Max, Min, Reverse, Take, TakeWhile, ToList, Union, Where.
A felsoroló (IEnumerable) objektumok[4] esetén használhatjuk a *sztringváltozó=változónév.metódusnév*.ToArray (!) metódushívást, vagy karaktertömbben tárolhatjuk az eredményt.
Lásd még: Függvények/Sztringkezelő függvények

[4] A Visual Basicben a sorozatok (tömb, lista stb.) elemeire vonatkozó több metódus úgynevezett felsoroló objektumot (pontosabban interfészt) eredményez. A felsoroló objektumot a Dim Változónév As IEnumerable([Of típus]) utasítással deklaráljuk, és a meghívott metódussal hozzuk létre, például: Felsorolás = Tömb.Distinct(). A metódus eredményét közvetlenül is átalakíthatjuk a megfelelő adatszerkezetre: Tömb2 = Tömb.Distinct().ToArray()

A String osztály osztálymetódusai

Névtér: System
Hivatkozás: String.*metódusnév*(*argumentumok*)

Két sztring összehasonlítása: String.Compare(*sztring1*[, *kezd1*], *sztring2*[, *kezd2*, *hossz*][, *mód*]) String.Compare(*sztring1*[, *kezd1*], *sztring2*[, *kezd2*, *hossz*], *hasonlítás*)	A *sztring1* és *sztring2* összehasonítása – a magyar ábécé szerint. *Mód* = True: nem különbözteti meg a kis- és nagybetűket. – a *hasonlítás* paraméter értéke szerint (lásd A sztringobjektum legfontosabb tulajdonságainál). Az összehasonítást a *kezd1*, illetve *kezd2* indexeknél kezdi, majd legfeljebb *hossz* darab karakteren keresztül folytatja. Függvényérték: –1 ha *sztring1* < *sztring2* 0 ha *sztring1* = *sztring2* 1 ha *sztring1* > *sztring2*
Formázott megjelenítés: String.Format(*sztring*, *érték0*, *érték1*, …)	A megadott értékeket formázott karaktersorozatként beágyazza a formátumokat tartalmazó *sztring* sztringbe. A formátum alakja (a kapcsos zárójel része a szintaxisnak): `{index[, hossz][:formátumkód]}`, melynek részei: *index*: a kiírásra kerülő érték sorszáma a paraméterlistában (0-val kezdődik a sorszámozás) *hossz*: az adott érték számára fenntartott karakterek száma pozitív érték: jobbra zár, negatív érték: balra zár *formátumkód*: a kiírásra kerülő érték formátuma. Formátumkód például F*n*: fixpontos megjelenítés *n* tizedesjeggyel; E*n*: normálalak *n* tizedesjeggyel. Például: `String.Format("A {0, 5:F1} négyzetgyöke: {1, 10:F4}.", 20, Math.Sqrt(20))`
Sztring szétvágása részekre: String.Split(*karaktertömb*[, *darab*])	A *sztring*-et a **karaktertömb** karaktereinél szétvágva elhelyezi egy sztring-tömbben. Legfeljebb *darab* részre vágja szét. A Split módosítja a sztingtömb méretét! Helyettesíti a tömbobjektum létrehozását és inicializálását.
Sztringtömb egyesítése egyetlen sztringgé: String.Join(*elválasztójel*, *sztringtömb*[, *kezdőindex*, *darab*])	A *sztringtömb darab* számú elemét a *kezdőindex*-től kezdve összefűzi egy sztringbe. Az elemek közé beilleszti az *elválasztójel* **sztringet**. Alapértelmezés: a teljes tömb.

A Strings osztály osztálymetódusai

Névtér: Microsoft.VisualBasic
A Strings osztály további osztálymetódusokat tartalmaz a sztringek kezeléséhez (például Asc, AscW, Chr, ChrW, InStr, InStrRev, Join, LCase, Left, Len, LTrim, Mid, Replace, Right, RTrim, Space, Split, StrComp, StrDup, StrReverse, Trim, UCase)

Strings.Split(*sztring*[, *elválasztójel*[, *darab*]])	A *sztring*-et az *elválasztójel* **sztringeknél** szétvágva elhelyezi egy sztring-tömbben. Legfeljebb *darab* részre vágja szét. A Split módosítja a sztringtömb méretét! Helyettesíti a tömbobjektum létrehozását és inicializálását.
Strings.Join(*sztringtömb*, *elválasztójel*]	A *sztringtömb* elemeit összefűzi egy sztringbe. Az elemek közé beilleszti az *elválasztójel* **sztringet**.

Megjegyzés: **a Split metódusnál sztringet is megadhatunk elválasztójelként**. Ügyeljünk az eltérő argumentumokra!

Változók és konstansok

Deklarálás, automatikus kezdőérték

Konstans deklarálása (programmodul, alprogram vagy blokkszinten)	`Const` *`név`* `[,` *`név`* `…] As` *`típus`* `=` *`érték`*`, …`	Egy értékhez több név rendelhető. Az *érték* fordítási időben kiértékelhető kifejezés is lehet. A függvények közül csak az Asc, AscW, Chr, ChrW szerepelhet a kifejezésben.

Változó deklarálása (programmodul, alprogram vagy blokkszinten)	`[`*`elérési mód`*`] [[Shared] [Static]] Dim …`
	
	`Dim` *`név`*`, … As` *`típus`*`, …`
	`Dim` *`név`* `As` *`típus`* `[=` *`kezdőérték`*`], …`
	`Dim` *`név`* `As New` *`konstruktor`*
		Kezdőérték megadása esetén a változónév után mindig ki kell írni a típust. A *kezdőérték* fordítási időben kiértékelhető kifejezés is lehet. Public, Private, Shared, Static megjelölés esetén elhagyható a Dim kulcsszó.

Automatikus kezdőérték	Numerikus változók: 0 Char-típus: bináris 0 Hivatkozás típusok: Nothing Boolean: False Date: #01/01/0001 12:00:00 AM#	objektumok, sztringek, tömbök stb.

Megjegyzés: a forráskódban tizedespontot, az adatbevitelnél a területi beállításoknak megfelelő elválasztójelet (például tizedesvesszőt) alkalmazunk.

A változók élettartama

Blokkszintű, illetve eljárásszintű (nem Static) változók élettartama: az alprogram futási ideje (az alprogramba való belépéstől az alprogramból való kilépésig tart).
- Az alprogramba való belépéskor a változó megkapja az automatikus értéket, amit a Dim kezdőértékadása vagy egy értékadó utasítás felülírhat.
- Ha az alprogram meghív egy másik alprogramot, akkor a változók megtartják közben értéküket (nem fejeződik be az élettartamuk).

Modulszintű változók élettartama
- Standard modulban deklarálva: a program futási ideje.
- Osztályban, illetve struktúrában deklarálva (nem Shared): megegyezik az adott típusú objektum, illetve változó élettartamával.

Megjegyzés: A blokkszintű változók inicializálása független attól, hogy az adott blokk végrehajtásra kerül-e. Élettartamuk az alprogram futásideje!!! Ha ismét belépünk a blokkba, megmarad az előző értékük! Ennek elkerüléséhez célszerű a blokkban inicializálni a változót (ne használjuk ki az automatikus kezdőértékadást)!

Az élettartam módosítása

Shared: az osztályok, illetve struktúrák megosztott (közös) változóinak/mezőinek minősítése. Ezek élettartama megegyezik a program futási idejével.
- A Shared változók nem kötődnek egy objektumhoz, illetve változóhoz. Az osztály/struktúra nevével minősítve hivatkozunk rájuk.
- Az objektumosztályok Shared tulajdonságait osztálytulajdonságnak vagy megosztott tulajdonságnak nevezzük.
- A struktúrák Shared mezőit megosztott tulajdonságnak vagy megosztott mezőnek nevezzük.

Static: meghosszabbítja az eljárás-, illetve blokkszintű változók élettartamát. A sztatikus változók élettartama:

Az alprogram helye	A statikus változó élettartamának	
	kezdete	vége
Programmodul (alapértelmezés: Shared eljárás)	Az alprogram első meghívása	A program futásának befejeződése.
Osztály, struktúra (nem Shared eljárás)	Az adott osztályhoz tartozó objektum vagy struktúra típusú változó alprogramjának első meghívása	Az objektum vagy változó felszabadítása (szemétgyűjtés).
Osztály, struktúra (Shared eljárás)	Az alprogram első meghívása (az osztály/sturktúra vagy egy objektum/változó nevével minősítve)	A program futásának befejeződése.

Sztatikus változó csak eljárás-, illetve blokkszinten deklarálható, de nem szerepelhet struktúra eljárásában.
A deklarációban szerepelő kezdőértéket csak az alprogram első meghívásakor veszi fel a sztatikus változó.
Az objektumosztályok Shared alprogramjainak sztatikus változói az osztályhoz tartoznak, csak egyetlen példányban léteznek a memóriában. A nem Shared alprogramok sztatikus változói az objektumpéldányhoz tartoznak, objektumonként külön-külön léteznek a memóriában (értékük objektumonként különbözhet).
Egy deklarációban nem szerepelhet egyszerre a Static és a Shared megjelölés.
Megjegyzés: Shared, illetve Static megjelölés esetén elhagyható a Dim kulcsszó a deklarációból.

Operátorok

A legfontosabb operátorok

		Megjegyzés
Aritmetikai műveletek	`+ - * / ^` `\ Mod`	Összeadás, kivonás, szorzás, osztás, hatványozás, maradékos osztás hányadosa, illetve maradéka.
Logikai műveletek	`Not, And, Or, Xor`	Nem, és, vagy, kizáró vagy
Logikai műveletek rövidzárral	`AndAlso, OrElse`	Ha az első operandusból következik az eredmény, akkor a második operandus nem kerül kiértékelésre.
Biteltoló műveletek (Aritmetikai eltolás: az eredmény előjele nem változik.)	*`operandus1`* `<<` *`operandus2`* *`operandus1`* `>>` *`operandus2`*	Az egész típusú 1. operandus bitjeit a 2. operandus által meghatározott mértékben eltolja balra, illetve jobbra. A 2. operandust a típushossznál eggyel kevesebb bitre maszkolja, így soha nem jön létre túlcsordulás.
Sztringek összefűzése	`&`	Az automatikus típuskonverzió miatt sztringet és numerikus értéket is összefűzhetünk (például a kiírásnál).
Értékmódosító műveletek	`^=, *=, /=, \=, +=, -=, &=`	Például: `A += B` egyenértékű `A = A + B`-vel
Relációk	`=, <>, <, >, <=, >=` `Is, IsNot`	Objektumhivatkozások összehasonlítása (Nothing-gel is!).

		Megjegyzés
Sztringek összehasonlítása	1. Relációs operátorokkal 2. Az StrComp függvénnyel: `StrComp(`*`sztring1`*`, `*`sztring2`*`, 1)` Függvényérték: –1 ha *sztring1* < *sztring2* 0 ha *sztring1* = *sztring2* 1 ha *sztring1* > *sztring2* 3. A Like operátorral: *`sztringkifejezés`* `Like` *`minta`* 4. A StringCompare osztálymetódussal	Ha az utolsó paraméter értéke 1, akkor az StrComp a magyar ábécé szerint végzi az összehasonlítást, de nem különbözteti meg a kis- és nagybetűket egymástól. Lásd még az Option Compare direktívát és a sztring osztály metódusait! Értéke True, ha a *sztringkifejezés* megfelel a *minta* sztringnek. A minta megfelel például az MS Access-ben használható mintáknak (*, ? #, [*karakterlista*], [!*karakterlista*]). Lásd A String osztály osztálymetódusainál.

Az operátorok precedenciája csökkenő sorrendben	`^`
	előjel (`+, –`)
	`*, /`
	`\`
	`Mod`
	`+, –`
	`&`
	`<<, >>`
	relációk, Is, IsNot, Like
	`Not`
	`And, AndAlso`
	`Or, OrElse`
	`Xor`
		Az azonos precedenciájú műveleteket a Visual Basic balról jobbra végzi el.

Utasítások

A legfontosabb utasítások

		Megjegyzés
Utasítás	Külön sorba írjuk. Szükség esetén folytatás a következő sorban: szóköz és aláhúzásjel a sor végére	Több utasítás egy sorban: kettősponttal elválasztva. A forráskódban nem különbözteti meg egymástól a kisbetűket és a nagybetűket.
Megjegyzés	`[`*utasítás*`] ' `*megjegyzés*	A sor végéig tart.
Értékadó utasítás	*változónév*`[(`*indexkifejezés* `[, …)]] = `*kifejezés*	

Feltételes elágazás	`If` *feltétel* `Then`
	  utasítások
	`[ElseIf` *feltétel* `Then`
	  utasítások`]`
	…
	`[Else`
	  utasítások`]`
	`End If`
	vagy:
	`If` *feltétel* `Then` *utasítások* `[Else` *utasítások*`]`
		Az Else-ág elmaradhat.
		
		Egysoros forma. Nincs End If.

Elágazás esetszétválasztással	`Select Case` *kifejezés*
	  `Case` *érték1*`[,` *érték2*`, …]`
	  utasítások
	  `[Case` *érték3*`[,` *érték4*`, …]`
	  utasítások`]`
	  `...`
	  `[Case Else`
	  utasítások`]`
	`End Select`
		Az első találathoz tartozó utasítások végrehajtása után kilép a Case szerkezetből. A Case Else ág elmaradhat. További lehetőségek:
		`Case` *alsóhatár* `To` *felsőhatár*: intervallum megadása, például: `Case 10 To 20`
		`Case Is` *relációjel kifejezés*: a relációnak megfelelő érték megadása, például: `Case Is <= 20`
		Lehet például:
		`Case 1 To 4, 7 To 9, 11, 13, Is > maxÉrték`
		Exit Select: kilép a Case szerkezetből.

Számlálós ciklus	`For` *számláló* `=` *kezdőérték* `To` *végérték* `_`
	  `[Step` *lépésköz*`]`
	  utasítások
	  `[Exit For]`
	  `[Continue For]`
	  utasítások
	`Next`
		A Step 1 elhagyható.
		A ciklusra nézve lokális ciklusváltozó deklarálása:
		`For` *számláló* `As` *típus* `=` *kezdőérték* …
		Visszafelé számlálós ciklus esetén a lépésköz negatív érték.
		Exit For: kilép a ciklusból.
		Continue For: a Next-nél folytatja a ciklust.
		Option Infer On esetén aláhúzással megjelöli, ha nem a ciklusfejben deklaráljuk a ciklusváltozót (nem rendel hozzá automatikusan típust).

		Megjegyzés

Iterátoros ciklus	`For Each` *iterátor* `[As` *típus*`] In {`*tömb* `|` *halmaz*`}`
	  utasítások
	`Next`
		Az iterátor sorra felveszi a tömbelemek értékét. Az elemek maguk nem módosíthatók, de hivatkozás típus esetén a tagok már igen.
		A ciklust kollekciókra (például listákra) is alkalmazhatjuk.

Elöltesztelő feltételes ciklus	`Do {While	Until}` *feltétel*   *utasítások*   `[Exit Do]`   `[Continue Do]`   *utasítások* `Loop`	While: ismétlési feltétel Until: kilépési feltétel Exit Do: kilép a ciklusból Continue Do: a Loop-nál folytatja a ciklust.

Hátultesztelő feltételes ciklus	`Do`
	  utasítások
	  `[Exit Do]`
	  `[Continue Do]`
	  utasítások
	`Loop {While
		Until}` *feltétel*	

Minősítőblokk	`With {`*objektum* `|` *struktúra*`}`
	  utasítások
	`End With`
		Egymásba ágyazott blokkok esetén csak a legbelső objektum azonosítója hagyható el a minősítésnél.

Vége	`End`	Lezárja a megnyitott fájlokat, törli a változókat és kilép a programból.
Megállás	`Stop`	Leállítja a programot, de nem zárja le a fájlokat és nem törli a változókat. A fejlesztőrendszeren belül történő futtatásnál megfelel egy töréspont elhelyezésének.

Beolvasás, kiírás

Részletesebben lásd a megfelelő objektumosztályok ismertetésénél.

		Megjegyzés
Beolvasás konzolalkalmazásban	*változónév* `= Console.ReadLine()`	Beolvasás a billentyűzetről.
Beolvasás Windows-alkalmazásban	Szövegdobozzal (Textbox)	Lásd később.
Beolvasás inputdobozzal	*változónév* `= InputBox( _` `  `*üzenet*`[, [`*ablakcím*`], `*kezdőérték*`])`	Konzol- és Windows-alkalmazásban is használható.
Sztring átalakítása numerikus értékké	*változónév* `= CInt(`*sztringkifejezés*`)` (vagy `CLng, CSng, CDbl` stb.)	Beolvasáskor mindig sztringet kapunk vissza!!! Automatikus típuskonverzió esetén nem kötelező konvertálni.

		Megjegyzés
Kiírás a képernyőre konzolalkalmazásban	`Console.Write(`*`sztringkifejezés`*`)` `Console.WriteLine(`*`sztringkifejezés`*`)`	Nem emel sort. Sort emel.
Kiírás a képernyőre Windows-alkalmazásban	Címkeobjektummal (Label)	Lásd később.
Kiírás üzenetablakkal	`MsgBox(`*`üzenet`*`[, [`*`gombok`*`], `*`ablakcím`*`])`	Konzol- és Windows-alkalmazásban is használható. *gombok*: például vbOkOnly
Numerikus érték formázása	`FormatNumber(`*`név`*`, `*`tizedesjegy`*`)`	Sztringet ad vissza a megadott tizedesjeggyel (lehet 0 is).
Képernyőtörlés konzolalkalmazásban	`Console.Clear()`	

Kivételkezelés

A Try utasítás szerkezete:

```
Try                                                          Próba-blokk
  [utasítások]
  [Exit Try]
[Catch[ kivétel[ As kivételtípus]][ When kifejezés]          Kivétel-blokkok
  [utasítások]
  [Exit Try]]
[Catch …]
[Finally                                                     Végül-blokk
  [utasítások]]
End Try
```

A próba-blokk az esetlegesen kivételhez vezető utasításokat tartalmazza. A kivétel-blokkokat a nekik megfelelő kivétel létrejöttekor hajtja végre a program. Egy kivétel csak a sorrendben első, neki megfelelő kivétel-blokk végrehajtását okozza. A végül-blokk utasításai a kivételkezelés után kerülnek sorra.
A *kivétel* egy Exception vagy belőle leszármazott típusú objektumot deklarál, melynek segítségével elérhetők a kivétel tulajdonságai. A kivételtípus hiányában az adott kivétel-blokk bármely kivételre vonatkozik. Az objektum Message tulajdonsága megadja a kivétel angol nyelvű leírását.
When megadása esetén a kivételblokk csak akkor kerül végrehajtásra, ha a *kifejezés* értéke True.
Exit Try esetén a végül-blokkban, ennek hiányában, illetve ez után pedig az End Try utasítást követő utasítással folytatódik a végrehajtás.
Egy kivétel létrejöttekor megszakad a próba-blokk további utasításainak a végrehajtása. A végül-blokk akkor is végrehajtásra kerül (ha van), ha nem jön létre kivétel.
Az egyes blokkokban deklarált változók blokkszintű hatókörrel rendelkeznek.
A Try utasításnak vagy kivétel-blokkot vagy végül-blokkot mindenképpen tartalmaznia kell.

Az Exception típusú objektumok tulajdonságai és metódusai

Message	A kivétel angol nyelvű leírása.
GetType()	A kivétel típusa (objektumosztálya).

Gyakoribb kivételtípusok (objektumosztályok)

Névtér: System

ArithmeticException	Hibás aritmetikai vagy konverziós művelet.
DivideByZeroException	Nullával való osztás (maradékos osztásnál).
Exception	Általános típusú kivétel.
FormatExeption	Hibás formátum.
InvalidCastException	Típuskonverziós hiba.
IO kivételek	Kivételek a fájlkezelésnél, lásd alább.
IndexOutOfRangeException	Indexhatár túllépése.
NullReferenceException	Hivatkozás nem létező objektumra.
OutOfMemoryException	Nincs elég memória a program végrehajtásához.
OverflowException	Túlcsordulás.
RankException	Hibás dimenziószám az alprogram tömbparaméterénél.
StackOverflowException	Túl sok egymásba ágyazott alprogramhívás (verem túlcsordulás).

Megjegyzés:
A *DivideByZeroException* (osztás nullával) a maradékos osztásnál (\) fordulhat elő. Ha a / műveleti jelet használjuk az egész típusú értékeknél, akkor a 0-val való osztás *OverflowException*-t (túlcsordulás kivételt) okoz. Valós típusú változóknál a Visual Basic az IEEE 754 szabvány alapján *Végtelen*-nek (*Infinity*) vagy *Nem szám*-nak (*Not a Number*) tekinti a 0-val osztás eredményét, így nem utasítja vissza a művelet elvégzését (nem jön létre kivétel). A *CInt* stb. típuskonverziós függvények alkalmazásakor *InvalidCastException* (tkp. számkonvertálás kivétel), a *Convert* objektumosztály osztálymetódusainak a használatakor (például *Convert.ToInt32*) pedig *FormatException* (tkp. hibás formátum kivétel) jöhet létre.

IO kivételtípusok

Névtér: System.IO

DirectoryNotFoundException	Nem érhető el a megadott mappa.
DriveNotFoundException	Nem érhető el a megadott meghajtó.
EndOfStreamException	Olvasási utasítás a fájl végét követően.
FileNotFoundException	Nem érhető el a megadott fájl.
PathTooLongException	Túl hosszú az elérési út sztringje.

Alprogramok

Minden végrehajtható utasításnak alprogramban kell elhelyezkednie.

Eljárások

Az eljárás szerkezete:

```
Sub eljárásnév([paraméterlista])
  [lokális deklarációk]
  [utasítások]
  [Exit Sub]
  [utasítások]
End Sub
```

A Sub utasítás csak modulszinten szerepelhet a programban. Ebből következik, hogy az eljárások nem ágyazhatók egymásba.
Az eljárások alapértelmezés szerint Public hozzáférésűek. Ezt az eljárásfejben módosíthatjuk.
Az Exit Sub utasítással kiugorhatunk az eljárásból.

Eseménykezelő eljárások

Az eseménykezelő eljárás szerkezete:

```
Sub eljárásnév(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles objektumnév.eseménynév, …
  [lokális deklarációk]
  [utasítások]
End Sub
```

Az *objektumnév* objektumon létrejött *eseménynév* eseményt kezeli
Az eljárásnév szokás szerint: *objektumnév_eseménynév*
Paraméterek:

sender: az eseményhez kapcsolódó objektum

e: eseményargumentum-objektum (hivatkozhatunk rá az eljárásban)

Egy eljáráshoz több esemény is megadható (a Handles után)
Eseménykezelő eljárás futásidejű hozzárendelése a vezérlőelemhez:

```
AddHandler vezérlőelem-objektum.esemény, AddressOf eseménykezelő eljárás neve
```

A hozzárendelésének megszüntetése:

```
RemoveHandler vezérlőelem-objektum.esemény, AddressOf eseménykezelő eljárás neve
```

Megjegyzés: eseménykezelő eljárás explicit módon (egy másik eljárásból) történő meghívásánál az eseményargumentum helyére írjuk be az üres eseményt (System.EventArgs.Empty).

Függvények

A függvény szerkezete

```
Function függvénynév([paraméterlista]) As visszatérési érték típusa
  [lokális deklarációk]
  [utasítások]
End Function
```

Az utasítások között egy vagy több helyen szerepelnie kell a

```
függvénynév = kifejezés
```

értékadásnak, ami a visszatérési értéket határozza meg.
A Function utasítás csak modulszinten szerepelhet a programban. Ebből következik, hogy a függvények nem ágyazhatók egymásba.
A függvények alapértelmezés szerint Public hozzáférésűek. Ezt a függvényfejben módosíthatjuk.
Az Exit Function utasítással kiugorhatunk a függvényből.
A függvények eljárásként is meghívhatók. Ebben az esetben figyelmen kívül marad a visszatérési érték.

Paraméterlista

A paraméterlista egyetlen paraméterből vagy egymástól vesszővel elválasztott paraméterekből áll. Egy paraméter szintaxisa:

```
[{ByVal | ByRef}] név[()] As típus
```

Tömb átadásánál az indexek jelölése nélkül tegyük ki a zárójelet: *tömbnév*().
A paraméterek az alprogramok lokális változóinak számítanak.
Megjegyzések.

A paramétereket alapértelmezett értékkel láthatjuk el. A paraméterlista tartalmazhat opcionális paramétereket, illetve paramétertömböt. Részletesebben lásd a Visual Basic súgóját.

Paraméterátadás cím és érték szerint

A ByVal érték szerinti, a ByRef cím szerinti paraméterátadást jelöl. Alapértelmezett a ByVal.
Ügyeljünk arra, hogy a hivatkozás-típusú paramétereknél (például tömböknél) ByVal esetén is módosítható az argumentum! Az argumentumként szereplő konstansok, literálok, felsorolások elemei illetve kifejezések természetesen ByRef esetén sem módosulnak.
A fordítóprogram a hatékonyság növelése érdekében átrendezheti az aritmetikai kifejezéseket. Ha az argumentumok függvényhívásokat tartalmaznak, nem számíthatunk az előírt sorrendben történő hívásra!
Megjegyzés: az argumentumok azonosítása történhet név és pozíció szerint. Részletesebben lásd a Visual Basic súgóját.

Függvényparaméterek

Függvényparaméterek esetén (lásd például a tömbmetódusokat) az argumentum helyére a függvény címét kell írni, amit az AddressOf operátorral határozunk meg:

```
AddressOf függvénynév
```

A függvény nevét zárójelek és argumentumok nélkül írjuk a kifejezésbe. Az argumentumok átadásának a módját lást a Súgóban (Invoke metódus).

Beépített függvények

A legfontosabb beépített függvények

Névtér: Microsoft.VisualBasic
Lásd még a Math osztály metódusait!
Lásd még: http://support.microsoft.com/kb/818805/hu

		Megjegyzés

Sztringkezelő függvények
Lásd még a sztring objektum tulajdonságait és metódusait!
	`Len(s), Left(s,db), Right(s,db), Mid(s,n,db)`
	
	`StrDup(ismétlés, "karakter")`
	`StrReverse(s)`
		A sztring hossza; balról, jobbról, illetve a megadott helytől kezdve a megadott számú karakter. Paraméterek *s*: sztringkifejezés, *db*: a karakterek száma, *n*: a kezdő karakter sorszáma.
		A *karakter*-t *ismétlés*-szer megismétli.
		Megfordítja a sztringet.

Karakterkódok és karakterek	`Asc("karakter"), AscW("karakter")` `Chr(kód), ChrW(kód)`	A karakter ANSI-kódját, illetve Unicode-ját adja vissza. Az ANSI-kód, illetve a Unicode alapján a karaktert adja vissza. Használatukhoz importálni kell a Strings névteret.
Típuskonverziós függvények	`CBol, CByte, CChar, CDate, CDbl, CDec, CInt, CLng, CSng, CStr`	Paraméter: a konvertálandó érték. Az eredmény típusa megfelel a függvénynévben szereplő típusnak. (Bol: Boolean, Dbl: Double, Int: Integer, Sng: Single, Str: String) Törtek egész értékre történő konvertálásánál (CByte, CInt, CLng) kerekíti a számot. **0,5 törtrész esetén a legközelebbi páros egészre kerekít** (bankár-kerekítés)!
Konvertálás numerikus értékre	`Val(sztringkifejezés)`	A sztringkifejezés elejét numerikus értékké konvertálja. **A sztringben tizedespontot kell használni!** Így a Val függvény közvetlenül nem alkalmas a szövegdobozzal beolvasott valós értékek konvertálására!
Beviteli ablak	`InputBox(szöveg[, cím[, érték[, x, y]]])`	Az *x*, *y* pozícióban elhelyezi a képernyőn a beviteli ablakot a megadott szöveggel, címmel és a szövegdobozba írt értékkel.

Egyéb függvények	`IsNumeric(kifejezés)`
	`IsDate(kifejezés)`
	`Hex(egész szám)`
	`Int(kifejezés)`
	`Fix(kifejezés)`
		A *kifejezés* értéke értelmezhető-e számként, illetve dátum/időként (True, False). Például beolvasás ellenőrzéséhez.
		Sztringként megadja a szám hexadecimális értékét.
		A numerikus kifejezés egészrésze.
		Elhagyja a numerikus kifejezés értékének törtrészét.

Megjegyzés: az IsNumeric(*logikai kifejezés*) értéke True (numerikus értékre konvertálható).

Típuskonverziós metódusok

Névtér: System.Convert
A típuskonverziót a Convert osztály alábbi osztálymetódusaival is elvégezhetjük:
ToBoolean, ToByte, ToChar, ToDateTime, ToDecimal, ToDouble, ToInt16, ToInt32, ToInt64, ToSingle, ToString.
Az egész típusnál az Integer helyett az Int rövidítést használjuk a bitmérettel kiegészítve (a .NET-nek megfelelő jelölésmód). A függvény argumentuma a konvertálandó kifejezés.
Megjegyzés: a legtöbb struktúra rendelkezik a *típus*.Parse(*sztringkifejezés*) sztatikus metódussal, amely a megadott típusra alakítja az argumentumsztringet.

A Math osztály tulajdonságai és metódusai

Névtér: System
A táblázat megosztott tulajdonságokat és sztatikus metódusokat tartalmaz. A Math minősítés elhagyható, ha a forráskód elején importjáljuk a System.Math névteret.
x, *y*: a függvénynek megfelelő típusú kifejezés

`Math.E`	Az *e* dupla pontosságú értéke.
`Math.Pi`	A π dupla pontosságú értéke.
`Math.Abs(`*x*`), Math.Sqrt(`*x*`)`	Abszolútérték, négyzetgyök.

`Math.Sin(`*x*`), Math.Cos(`*x*`), Math.Tan(`*x*`)`
`Math.Acos(`*x*`), Math.Asin(`*x*`), Math.Atan(`*x*`)`
`Math.Atan2(`*y*`, `*x*`)`
	Szögfüggvények (radián argumentummal!)
	A szögfüggvényből visszaszámolja a szöget (radiánban).
	Az y/x arkusz tangense (értelmezi az *x* = 0-t is).

`Math.Exp(`*x*`), Math.Log(`*x*`)` `Math.Log10(`*x*`)`	*e* alapú hatvány, logaritmus 10-es alapú logaritmus
`Math.Ceiling(`*x*`)`	A legkisebb egész, amely nagyobb vagy egyenlő az *x*-nél.
`Math.Floor(`*x*`)`	A legnagyobb egész, amely kisebb vagy egyenlő az *x*-nél.
`Math.Max(`*x*`, `*y*`), Math.Min(`*x*`, `*y*`)`	A két érték maximuma/minimuma.
`Math.Pow(`*x*`, `*y*`)`	x^y (hatványozás)
`Math.Round(`*x*`[, `*n*`])`	*n* tizedesre kerekít (kerekítés egészre: *n* = 0).
`Math.Sign(`*x*`)`	Előjelfüggvény (x<0 esetén –1; x = 0 esetén 0; x>0 esetén +1)
`Math.Truncate(`*x*`)`	Elhagyja az *x* törtrészét.

Összetett típusok

Tömbök

Névtér: System
A tömbök objektumok! Hivatkozás-típusú változó mutat rájuk. A *tömb2 = tömb1* értékadás a 2. tömbre mutató hivatkozást átállítja az 1-es tömbre, így mindkét változó ugyanarra a tömbre fog mutatni! (A program a 2-es tömböt törli a memóriából, ha más hivatkozás nem mutat rá.). A tömb duplikálásához az értékadás helyett használjuk a CopyTo metódust!

		Megjegyzés

Tömb deklarálása	`Dim tömbnév(maxindex[, …]) As elemtípus`
	`Dim tömbnév([, …]) As elemtípus = _`
	`{ elemek felsorolása }`
	vagy:
	`Dim tömbnév([, …]) As elemtípus`
	`tömbnév = New elemtípus(maxindex[, …]) {}`
		Maxindex: az index legnagyobb értéke.
		Egy tömbnek legfeljebb 32 indexe lehet.
		A tömbelemek indexelése mindig 0-val kezdődik!
		A kapcsos zárójel a szintaxis része!
		
		A kapcsos zárójel itt kötelező akkor is, ha nem soroljuk fel a kezdőértékeket! A New helyett használhatjuk a ReDim utasítást.

Kezdőértékadás több dimenzió esetén	például: `{{1, 2, 3}, {4, 5, 6}}`	
Hivatkozás a tömb elemeire	`tömbnév(indexkifejezés[, …])`	Indexkifejezés: konstans, változónév, kifejezés (egészre kerekít!)
Az *i*-edik dimenzió maximális indexe	`tömbnév.GetUpperBound(i)`	A dimenziók számozása 0-val kezdődik.

A tömbméret módosítása	`ReDim [Preserve] tömbnév(újmaxindex1, …)`
	
	vagy:
	`Array.Resize(tömbnév, újelemszám)`
		A program készít egy új tömböt, és Preserve esetén átmásolja a régi tömb elemeit az újba. Az elemek értékének megőrzése (Preserve) esetén csak az utolsó dimenzió mérete módosítható!
		A Resize metódus csak egydimenziós tömbökre alkalmazható. Megőrzi a tömb elemeinek értékét. Argumentumként az új tömb elemszámát kell megadni (nem pedig a maximális indexet)!

A tömb felszabadítása	`Erase tömb1[, tömb2, …]`	Felszabadítja a tömb által lefoglalt memóriát, és Nothing-ra állítja a változót.

Megjegyzés: a többi objektummal ellentétben a tömb létrehozásánál a New operátor után nem az objektum konstruktorát hívjuk meg, csupán a tömbelemek típusát jelezzük. A nem publikus konstruktor meghívása helyett szükség esetén használhatjuk a CreateInstance osztálymetódust (futás alatti kötés).

Hatékonyabb programot írhatunk, ha tömb helyett listát használunk. Különösen a tömbméret módosítása vesz el sok időt.

A tömbobjektum tulajdonságai és metódusai

Hivatkozás: *tömbnév.tulajdonságnév*, *tömbnév.metódusnév*(*argumentumok*)

Length	A tömbelemek száma, az összes dimenziót beleértve.
Rank	A tömb dimenzióinak a száma.
CopyTo(*tömb2*, *index*)	Az **egydimenziós** tömb összes elemének átmásolása az egydimenziós *tömb2*-be a *tömb2*(*index*)-től kezdve.
GetUpperBound(*i*)	Megadja az *i*-edik index legnagyobb értékét (0-val kezdődik az indexek sorszámozása).

Az Array osztály osztálymetódusai

Hivatkozás: Array.*metódusnév*(*argumentumok*)

Array.BinarySearch(*tömb*[*kezdőindex*, *darab*], *elem*[,*komparálófv*])	Bináris kereséssel megkeresi az **egydimenziós, rendezett** *tömb* tömbben az *elem* elemet. A keresést a kezdőindex-től kezdi és darab elemen keresztül folytatja (alapértelmezés: az egész tömb). A visszatérési érték a megtalált elem indexe (negatív értéket ad vissza, ha nem találta meg). A komparálófüggvény használatát lásd a tankönyv kiegészítésében!
Array.Clear(*tömb*, *kezdőindex*, *elemszám*)	A *tömb* tömb *elemszám* darab elemének törlése a *kezdőindex*-től kezdve. A törlés után a tömb elemeinek értéke: 0 (numerikus), False (logikai), Nothing (objektum). Eljárásként kell meghívni!
Array.CreateInstance(*típus*, *méret1*[, *méret2*, …])	Létrehozza a tömböt a késői kötéshez.
Array.Exists(*tömb*, AddressOf *predikátumfv*)	Lineáris kereséssel meghatározza, hogy létezik-e az **egydimenziós** *tömb*-ben a *predikátumfüggvény* által megadott elem (True/False). A predikátumfüggvény értelmezését lásd a 29. oldalon!
Array.Find(*tömb*, AddressOf *predikátumfv*)	Lineáris kereséssel meghatározza az **egydimenziós** *tömb*-ben a predikátumfüggvénnyel kiválasztott elemet. Ha nem talál megfelelő értéket, akkor az elem alapértelmezett értékével tér vissza.
Array.FindAll	Ugyanaz, mint az Array.Find, de egy tömböt ad vissza, amely tartalmazza az összes megfelelő elemet. Ha nem talál megfelelő értéket, akkor üres tömböt ad vissza.
Array.FindIndex(*tömb*[, *kezdőindex*[, *darab*]], AddressOf *predikátumfv*)	Ugyanaz, mint az Array.Find, de az elem indexét adja vissza. A keresést a *kezdőindex*-nél kezdi és *darab* elemen át folytatja (alapértelmezés: az egész tömb).
Array.FindLast, Array.FindLastIndex	Ugyanaz mint az Array.Find, illetve az Array.FindIndex, de a keresést a tömb végén kezdi.
Array.ForEach(*tömb*, AddressOf *transzformációsfv*)	Az **egydimenziós** *tömb* elemeire végrehajtja a transzformációs függvényt. A transzformációs függvény értelmezését lásd a 29. oldalon. Eljárásként kell meghívni!
Array.IndexOf(*tömb*, *elem*[, *kezdőindex*[, *elemszám*])	Az *elem* elem lineáris keresése az **egydimenziós** *tömb*-ben a *kezdőindex*-től kezdve *elemszám* darab elemen keresztül (alapértelmezés: az egész tömb). A visszatérési érték az első megtalált elem indexe (–1 ha nem találta meg).
Array.LastIndexOf	Ugyanaz, mint az Array.IndexOf, csak visszafelé keres (a tömb végétől kezdve).

Array.Reverse(*tömb*[, *kezdőindex*, *elemszám*])	Megfordítja az **egydimenziós** *tömb* tömb *elemszám* darab elemének sorrendjét a *kezdőindex*-től kezdve (alapértelmezés: az egész tömb).
Array.Sort(*tömb*[, *kezdőindex*, *elemszám*] [, *komparálófv*])	Rendezi az **egydimenziós** *tömb* tömb *elemszám* darab elemét a *kezdőindex*-től kezdve. A komparálófüggvény értelmezését lásd a 29. oldalon!
Array.Sort(*kulcstömb*, *értéktömb* …)	Ugyanaz, mint az Array.Sort, de a *kulcstömb* elemeivel együtt rendezi a hozzájuk tartozó, egy dimenziós *értéktömb* elemeit is.

További egy dimenziós tömbmetódusok

Hivatkozás: *tömbnév*.*metódusnév*(*argumentumok*)

All(AddressOf *predikátumfv*)	A tömb összes eleme rendelkezik-e a predikátumfüggvény által meghatározott tulajdonsággal (True/False).
Any(AddressOf *predikátumfv*)	Van-e olyan tömbelem, amely rendelkezik a predikátumfüggvény által meghatározott tulajdonsággal (True/False).
Average([AddressOf *transzformációsfv*])	A (transzformációs függvény által módosított) tömbelemek átlaga.
Concat(*tömb2*)	Visszatérési értéke a tömbhöz hozzáfűzött *tömb2*, mint felsoroló (IEnumerable) objektum.
Contains(*érték*[, *komparálófv*])	A tömb tartalmazza-e a megadott értéket (True/False).
Count([*predikátumfv*])	A predikátumfüggvénynek megfelelő tömbelemek száma (alapértelmezés: a tömbelemek száma).
Distinct()	Felsoroló (IEnumerable) objektum, amely az egymástól különböző elemeket tartalmazza.
Except(*tömb2*)	Felsoroló (IEnumerable) objektum, amely a tömb azon, egymástól különböző elemeit tartalmazza, melyek nincsenek benne a *tömb2*-ben.
Intersect(*töm2*)	Felsoroló (IEnumerable) objektum, amely a két tömb közös elemeit tartalmazza (halmazfelsorolásként).
Max([AddressOf *transzformációsfv*]), Min([AddressOf *transzformációsfv*])	A (transzformációs függvény által módosított) tömbelemek maximuma, minimuma.
Sum([AddressOf *transzformációsfv*])	A (transzformációs függvény által módosított) tömbelemek összege.
Take(*Db*)	A tömb első *Db* számú elemét tartalmazó felsoroló (IEnumerable) objektum.
TakeWhile(AddressOf *predikátumfv*)	Azon tömbelemek összefüggő sorozat a tömb elejétől kezdve, melyek megfelelnek a predikátumfüggvénynek (IEnumerable objektum).
ToList()	Visszatérési értéke a tömbelemekből álló listaobjektum.
Union(*tömb2*)	A két tömb unióját tartalmazó felsoroló (IEnumerable) objektum (halmazfelsorolás).
Where(AddressOf *predikátumfv*)	A predikátumfüggvénynek megfelelő tömbelemeket tartalmazó felsorló (IEnumerable) objektum.

Megjegyzés: A Visual Basicben a sorozatok[5] (tömb, lista stb.) elemeire vonatkozó több metódus úgynevezett felsoroló objektumot (pontosabban interfészt) eredményez. A felsoroló objektumot a Dim Változónév As IEnumerable([Of típus]) utasítással deklaráljuk, és a meghívott metódussal hozzuk létre, például: Felsorolás = Tömb.Distinct(). A metódus eredményét közvetlenül is átalakíthatjuk a megfelelő adatszerkezetre: Tömb2 = Tömb.Distinct().ToArray()

Függvényparaméterek a tömbmetódusoknál

A függvényparaméterekkel rendelkező tömbmetódusok a tömb elemeit egyesével átadják a függvényparaméternek, majd a visszatérési értéket használják fel a végeredmény meghatározásához.

Predikátumfüggvények

Visszatérési értékük True vagy False.

```
Function függvénynév(paraméter As tömbelemtípus) As Boolean
  …
End Function
```

Transzformációs függvények

A tömbelemből képezett értékkel térnek vissza.

```
Function függvénynév(paraméter As tömbelemtípus) As típus
  …
End Function
```

Szelektorfüggvények

A transzformációs függvényt szokás szelektorfüggvénynek nevezni, ha egy struktúra (rekord) egy mezőjét adja vissza:

```
Function függvénynév(paraméter As struktúratípus) As mezőtípus
  …
End Function
```

A szelektorfüggvényt általában akkor alkalmazzuk, ha egy tömb struktúra típusú elemeket tartalmaz, de a tömbmetódusnak a sturktúra egy mezőjére van szüksége.
Megjegyzés: mivel a függvények alapértelmezés szerint Public hozzáférésűek, ezért a struktúrát, illetve a struktúra definícióját tartalmazó modult is lássuk el Public hozzáféréssel, vagy pedig a függvénynek írjunk elő Private hozzáférési módot!

Hasonlító (komparáló) függvények

A tömb két elemét hasonlítják össze. Két paraméterének típusa megegyezik a tömbelemek típusával:

```
Function függvénynév(param1 As tömbelemtípus, param2 As tömbelemtípus) As Integer
  …
End Function
```

A komparáló függvény visszatérési értéke: < 0 ha param1 < param2 = 0 ha param1 = param2 > 0 ha param1 > param2

[5] Pontosabban kollekciók.

Struktúrák (rekordok)

A struktúrát projekt- vagy modulszinten kell definiálni. A definíció szintaxisa:

```
Structure név
  meződeklarációk
  [metódusdeklarációk]
End Structure
```

Meződeklarációk: Const, Dim (továbbá: Enum, Events) utasítások
Metódusdeklarációk: Sub … End Sub, Function … End Function (továbbá: Operator, Property). A metódusok között megadhatunk konstruktort is.
A struktúra definíciójában nem adhatunk kezdőértéket a mezőknek (a megosztott tulajdonságok kivételével). Tömb típusú mezők esetén például a deklarációban nem adhatjuk meg a tömb méretét. Ezt a struktúra-típusú változó deklarációja után kell megtennünk (Redim vagy New).
A struktúra-típusú változók deklarálása magába foglalja a konstruktor meghívását:
A `Dim változónév As struktúranév` egyenértékű a `Dim változónév As struktúranév = New konstruktor()` utasítással
A struktúra-típusú változók mezőinek kezdőértékét megadhatjuk közvetlenül értékadással, vagy létrehozásakor a With operátorral:

```
Dim változónév As struktúranév
változónév = New struktúranév With {.mezőnév1 = érték1, .mezőnév2 = érté2k, …}
```

vagy összevonva:

```
Dim változónév As New struktúranév With {.mezőnév1 = érték1, .mezőnév2 = érté2k, …}
```

A struktúra tagjaira (mezők, metódusok) a struktúra-típusú változó nevével minősítve hivatkozunk:

```
változónév.tagnév[(argumetumok)]
```

A struktúra típusú változók érték-típusúak. Így a *változó2 = változó1* értékadás átmásolja az 1. változó adattagjainak értékét a 2. változó adattagjaiba (a hivatkozás-típusú tagok, például tömbök esetén természetesen csak a hivatkozást).
A struktúráknál nincsen öröklődés.
Megjegyzés: rekordokat struktúrákban tárolhatunk. A struktúrák mezőit szokás adattagoknak is nevezni.

Halmazok

A halmaz olyan kollekció, amely nem tartalmaz ismétlődő elemeket és nem értelmezzük az elemek sorrendjét.

Névtér: System.Collections.Generic
Halmaz deklarálása és létrehozása:

```
Dim változónév As Hashset(Of típus)
változónév = New HashSet(Of típus)[(kollekció)]
```

ahol a kollekció helyére tömböt, listát vagy bármilyen, felsoroló (IEnumerable) objektumot írhatunk. Megadása esetén a program átmásolja a halmazba a kollekció elemeit, kihagyva az ismétlődéseket. A kollekciót felhasználhatjuk a halmaz inicializálására.
A Visual Basic 2010-es változatában a létrehozás során megadhatjuk a halmaz elemeit:

```
Dim változónév As HashSet(Of típus)
változónév = New HashSet(Of típus) From {kezdőértékek felsorolása, vesszővel elválasztva}
```

Megjegyzés: ha összevonjuk a deklarálást és a létrehozást, akkor nem használhatjuk a kibővített metódusokat (All, Any, Max stb.).

A halmaz elemeire a feltöltés sorrendjében indexükkel is hivatkozhatunk: *változónév*.ElementAt(*index*) vagy: *változónév*(*index)*. Az indexelés 0-val kezdődik.

A halmazobjektum tulajdonságai és metódusai

A tulajdonságok, metódusok futásidejére vonatkozóan általában lásd a Visual Basic súgóját!
A halmazműveleteket az első operandus metódusaival végezzük, melyek argumentuma a második operandus (halmaz). A művelet eredménye az első operandusba kerül, például az A.UnionWith(B) metódushívás eredményeként A = A ∪ B lesz.

Count	A halmazelemek száma.
Add(*elem*)	Hozzáadja az elemet a halmazhoz. Visszatérési értéke True, ha megtörtént a hozzáadás (azaz az elem még nem szerepelt a halmazban), egyébként pedig False.
Clear()	Törli a halmaz elemeit. A memóriában fenntartott helyet a TrimExcess metódussal csökkenthetjük.
Contains(*elem*)	True, ha a halmaz tartalmazza a megadott elemet, egyébként pedig False. A futásidő független az elemszámtól!
CopyTo(*tömb*[, *index*[, *darab*]])	Átmásolja a halmaz *darab* számú elemét a *tömb*-be, a *tömb* megadott indexétől kezdve. Alapértelmezés a halmaz összes eleme a 0 indextől (a *tömb* elejétől) kezdve.
ExceptWith(*halmaz2*)	A *halmaz2* elemeit kivonja a halmazból. A művelet módosítja a halmazt (ide kerül az eredmény).
IntersectWith(*halmaz2*)	Meghatározza a halmaz metszetét a *halmaz2* halmazzal. A művelet módosítja a halmazt (ide kerül az eredmény).
IsProperSubsetOf(*halmaz2*)	True, ha a halmaz valódi részhalmaza a *halmaz2*-nek.
IsProperSupersetOf(*halmaz2*)	True, ha a *halmaz2* valódi részhalmaza a halmaznak.
IsSubsetOf(*halmaz2*)	True, ha a halmaz részhalmaza a *halmaz2*-nek.
IsSupersetOf(*halmaz2*)	True, ha a *halmaz2* valódi részhalmaza a halmaznak.
Remove(*elem*)	Törli az elemet a halmazból. Visszatérési értéke True, ha megtörtént a törlés (azaz az elem szerepelt a halmazban), egyébként pedig False.
SetEquals(*halmaz2*)	True, ha a halmaz és a *halmaz2* ugyanazokból az elemekből állnak, egyébként pedig False. Futásidő: az elemek számával arányos.
SymmetricExceptWith(*halmaz2*)	A halmaz és a *halmaz2* szimmetrikus különbsége (unió – metszet). A művelet módosítja a halmazt (ide kerül az eredmény).
ToArray()	Visszatérési értéke a halmaz elemeiből képezett tömb.
TrimExcess()	Az elemek számának megfelelő mértékben módosítja a memóriafoglalás méretét.
UnionWith(*halmaz2*)	Meghatározza a halmaz unióját a *halmaz2* halmazzal. A művelet módosítja a halmazt (ide kerül az eredmény).

Az ExceptWith, IntersectWith, IsProperSubsetOf, IsProperSupersetOf, IsSubsetOf, IsSupersetOf, SetEquals, SymmetricExceptWith metódusok argumentuma halmaz helyett bármely más, felsoroló objektum is lehet. A metódusok nem veszi figyelembe az argumentumként megadott objektumban az elemek ismétlődését, illetve sorrendjét.

A következő metódusok ismertetése a tömböknél található: All, Any, Average, Max, Min, Reverse, Sum, Take, TakeWhile, Where.

Rendezett halmazok (kollekciók)

Rendezett halmazokban olyan sorozatokat (kollekciókat) tárolhatunk, melyeknek nincsenek egyforma elemeik. Az elemek hozzáadáskor rendezve kerülnek a halmazba. A rendezett halmazok a műveletek szempontjából halmazként viselkednek, de elemeiket rendezve tárolják, és rendezve érjük el (indexelés, listázás).

Névtér: System.Collections.Generic
Rendezett halmaz deklarálása és létrehozása:

```
Dim változónév As SortedSet(Of típus)
változónév = New SortedSet(Of típus)[(kollekció)]
```

ahol a kollekció helyére tömböt, listát vagy bármilyen, felsoroló (IEnumerable) objektumot írhatunk. Megadása esetén a program rendezett sorozatként átmásolja a halmazba a kollekció elemeit, kihagyva az ismétlődéseket. A kollekciót felhasználhatjuk a halmaz inicializálására.
A Visual Basic 2010-es változatában a létrehozás során megadhatjuk a halmaz elemeit:

```
Dim változónév As SortedSet(Of típus)
változónév = New SortedSet(Of típus) From {kezdőértékek felsorolása, vesszővel elválasztva}
```

Megjegyzés: ha összevonjuk a deklarálást és a létrehozást, akkor nem használhatjuk a kibővített metódusokat (All, Any, Max stb.).
A rendezett halmaz elemeire indexükkel is hivatkozhatunk: *változónév*.ElementAt(*index*) vagy: *változónév*(*index)*. Az indexelés 0-val kezdődik.
A rendezett halmazok tulajdonságai és metódusai megegyeznek a halmaz objektumosztály fent felsorolt tagjaival.

Halmazműveletek megvalósítása tömbökkel

Esetenként szükség lehet tömbök (listák stb.) halmazként való kezelésére. A tömbökkel halmazműveleteket végezhetünk. A műveletek végzése előtt célszerű a tömbből halmazfelsorolást készíteni, de a legtöbb halmazművelet eredménye szintén halmazfelsorolás lesz.

Halmazfelsorolás készítése

Halmazfelsorolás: olyan sorozat (tömb, lista stb.), amely nem tartalmaz egyforma elemeket.
Halmazfelsorolás létrehozása: Tömb = Tömb.Distinct().ToArray()

Tömbök kezelése halmazként

Az elemek száma: Tömb.Count()
A megadott *Elem* benne van-e a halmazban (True/False): Tömb.Contains(*Elem*)

Halmazműveletek halmazként kezelt tömbökkel

A halmazműveleteket az egyik operandus metódusával végezzük el, melynek argumentuma a művelet másik operandusa.
A műveletek eredménye felsoroló (IEnumerable) objektum, melyet például a ToArray metódussal alakíthatunk vissza tömbbé.
Unió: *Unióhalmaz* = *Halmaz1*.Union(*Halmaz2*).ToArray()
Metszet: *Metszethalmaz* = *Halmaz1*.Intersect(*Halmaz2*).ToArray()
Különbség (Halmaz1 – Halmaz2): *Különbséghalmaz* = *Halmaz1*.Except(*Halmaz2*).ToArray()

Részhalmazok, halmazok egyenlősége

A részhalmaz vizsgálatát például tömbmetódusokkal (All, Any) vagy ciklussal végezhetjük el (eldöntés), illetve halmazműveletekre vezethetjük vissza.
Vizsgálat halmazműveletekkel:
H1 $\subseteq$ *H2* akkor és csak akkor, ha: *H1*.Count = *H1*.Intersect(*H2*).Count.
H1 $\subset$ *H2* (valódi részhalmaz) akkor és csak akkor, ha *H1*.Count < *H2*.Count és *H1* részhalmaza *H2*-nek (lásd előbb).
Két halmazfelsorolás egyenlőségét például tömbmetódusokkal (All, Any) vagy ciklussal vizsgálhatjuk meg (eldöntés), illetve halmazműveletekre vezethetjük vissza.
Vizsgálat halmazművelettel:

H1 = *H2* akkor és csak akkor, ha: *H1*.Count = *H2*.Count és *H1* részhalmaza *H2*-nek (lásd előbb)
Megjegyzés: az ellenőrzéshez nem kell visszaalakítani a felsoroló (IEnumerable) objektumot tömbbé.

Listák

A Visual Basic List objektumosztálya a dinamikus tömböket reprezentálja. Objektumaival hatékonyan kezelhetünk tömböket, főleg akkor, ha szükség van a tömbméret módosítására. A klasszikus értelemben vett listákat (láncolt lista, LinkedList) a tankönyv 2. kötetében mutatjuk be.

Névtér: System.Collections.Generic
A listák objektumok. Lista deklarálása és létrehozása:

```
Dim változónév As List(Of típus)
változónév = New List(Of típus)
```

A Visual Basic 2008-as változatában közvetlenül nem inicializálhatjuk a listaelemeket. Szükség esetén inicializáljunk egy tömböt, majd a ToList metódussal alakítsuk listává.
A Visual Basic 2010-es változatában a létrehozás során megadhatjuk a listaelemek kezdőértékét:

```
Dim változónév As List(Of típus)
változónév = New List(Of típus) From {kezdőértékek felsorolása, vesszővel elválasztva}
```

Megjegyzés: ha összevonjuk a deklarálást és a létrehozást, akkor nem használhatjuk a kibővített metódusokat (All, Any, Max stb.).
Hivatkozás a lista egy elemére: *változónév*.Item(*index*) vagy: *változónév*(*index*). Az indexelés 0-val kezdődik.
A listaelemek közvetlenül módosíthatók és lekérdezhetők: *érték* = *változónév*.Item(*index*); *változónév*.Item(*index*) = *érték*. A listaelemek elérési ideje független a lista elemszámától! A tulajdonságok, metódusok futásidejére vonatkozóan általában lásd a Visual Basic súgóját!

A listaobjektum tulajdonságai és metódusai

Capacity	A lista kapacitása, melyen belül nem szükséges a lefoglalt memóriaterület bővítése új elem hozzáadása esetén. (A memóriaterület bővítése automatikusan történik.)
Count	A listaelemek száma.
Item(*index*)	Az *index* indexű listaelem.
Add(*elem*)	Hozzáfűzi az *elem*-et a lista végéhez.
Clear()	Törli a lista elemeit (Count = 0). Futásidő: egyenesen arányos az elemszámmal.
Contains(*elem*)	Tartalmazza-e a lista a megadott elemet (True/False).
Insert(*index*, *elem*)	Beilleszti az *elem*-et az *index* helyre. Futásidő: egyenesen arányos az elemszámmal.
Remove(*elem*)	Törli az *elem* első előfordulását a listából. Futásidő: egyenesen arányos az elemszámmal.
RemoveAll(AddressOf *predikátumfv*)	A predikátumfüggvény által meghatározott elemek törlése. Futásidő: egyenesen arányos az elemszámmal.
RemoveAt(*index*)	A megadott indexű elem törlése. Futásidő: egyenesen arányos az elemszámmal.
ToArray()	Visszatérési értéke a listaelemekből képezett tömb.

A következő metódusok ismertetése a tömböknél található: All, Any, Average, BinarySearch (rendezett listában), Distinct, Except, Exists, Find, FindAll, FindIndex, FindLast, FindLastIndex, ForEach, IndexOf, Intersect, LastIndexOf, Max, Min, Reverse, Sort, Sum, Take, TakeWhile, Union, Where.
A felsoroló (IEnumerable) objektumok a ToList() metódussal alakíthatók listává.

A grafikus felhasználói felület kezelése

Vezérlőelemek

Névtér: System.Windows.Forms
Hivatkozás az objektumpéldányra az osztálydefiníción belül: `Me`

A legtöbb objektumnál előforduló tulajdonságok (a lehetséges értékeket lásd a *Properties* munkaablakban)			
Name	az objektum azonosítója	Left, Top	a bal felső sarok pozíciója a magábafoglaló objektumhoz viszonyítva
Anchor, Dock	az objektum helyzetének/méretének rögzítése a szülő vezérlőelemhez képest	Opacity	átlátszatlanság (százalékban)
		TabIndex	bejárási sorrend (tabulátorral)
AutoSize	automatikus méretezés (Label!) engedélyezése	TabStop	részt vesz-e a tabulátorral történő bejárásban
BackColor	háttérszín	Tag	tetszőleges érték tárolására fenntartott tulajdonság (például felhasználható a vezérlőelem azonosítására)
BackgroundImage	háttérkép (elérési út)		
BorderStyle	a szegély típusa	Text	a megjelenő szöveg
Cursor	a kurzor ikonja (lásd az intelligens súgót)	TextAlign	a szöveg igazítása
Enabled	engedélyezett-e a működése	Visible	látható-e az objektum
Font	a szöveg tulajdonságai	Width, Height	szélesség és magasság pixelben
Forecolor	betűszín		
Image	a vezérlőelemen megjelenő kép		

Megjegyzés: Az Anchor és a Dock tulajdonságok közül csak az egyik adható meg (az utoljára végzett módosítás lesz érvényes). Futásidőben egyszerre több irány logikai műveletekkel állítható be, például: `AnchorStyles.Bottom Or AnchorStyles.Right`.

A vezérlőelemeknél előforduló legfontosabb metódusok			
BringToFront()	előrehozza az elemet	Refresh()	érvényteleníti az elem megjelenítését, közvetlenül újrarajzolja az elemet és a gyermekelemeket
Focus()	a fókusz átvétele		
Hide()	elrejti az elemet	SendToBack()	hátrateszi az elemet
Invalidate()	érvényteleníti az elem megjelenését (újrarajzolást von maga után), szinkron módon történő rajzoláshoz hívjuk meg utána az Update metódust!	Show()	megjeleníti az elemet

A legfontosabb objektumok

A gyakran előforduló tulajdonságokat lásd fent!

	Osztály	További tulajdonságok, metódusok
Ablak (Űrlap)	Form	AcceptButton az Enter lenyomása a kijelölt gombra kattintással egyenértékű AutoScroll görgetősávok automatikus megjelenítése (szükség esetén) CancelButton az Esc lenyomása a kijelölt gombra kattintással egyenértékű ControlBox a rendszermenü megjelenítése FormBorderStyle a szegély típusa Icon a program ikonja MaximizeBox, MinimizeBox méretezőgombok megjelenítése MaximumSize.Width, Height az ablak maximális mérete MinimumSize.Width, Height az ablak minimális mérete ShowInTaskbar megjelenjen-e a tálcán a program StartPosition a megjelenítés helye a képernyőn Text a címsor szövege WindowState megjelenítés módja az ablak indításánál Close() az ablak bezárása Controls.Add(*vezérlőelem-objektum*) a vezérlőelem hozzáadása a Controls kollekcióhoz (megjelenik az ablakban) a vezérlőelemet először létre kell hozni!
Címke	Label	TextAlign a szöveg igazítása
Képdoboz	PictureBox	Image egy Image-objektum hozzárendelése a képdobozhoz (lásd az Image objektumosztály leírásánál). ImageLocation a kép elérési útja (megadásakor betölti a képet). Tervezésidejű megadásakor ne tegyük idézőjelbe az elérési utat (a Properties munkaablakban)! SizeMode méretezés (Normal: a kép vágása a képdoboznak megfelelően, StretchImage: a kép torzítása a képdoboznak megfelelően, AutoSize: a képdoboz méretezése a kép méretének megfelelően, CenterImage: ugyanaz mint a Normal, csak középről vág, Zoom: a kép arányos méretezése a képdoboznak megfelelően). CreateGraphics() grafikaobjektum létrehozása és hozzárendelése a képdobozhoz Load(*elérési út*) betölti és megjeleníti a megadott képet
Magyarázódoboz	ToolTip	ToolTipTitle a doboz címsorának szövege SetToolTip(*vezérlőelem, szöveg*): hozzárendeli a dobozt a megadott vezérlőelemhez a megadott szöveggel.
Parancsgomb	Button	Billentyűparancs hozzárendelése a parancsgombhoz: a Text tulajdonságban egy karakter elé & jelet írunk (elérés: Alt + karakter).

Szövegdoboz	TextBox	CharacterCasing	kis- vagy nagybetűk jelenjenek-e meg (Normal, Upper, Lower)
		Lines	többsoros szövegdoboz esetén a sorokat tartalmazó sztringtömb
		MaxLength	a beírható szöveg maximális hossza
		Multiline	engedélyezett-e több sor bevitele
		PasswordChar	a beírt karakterek helyett a megadott karakterek megjelenítése
		ReadOnly	csak olvasható-e
		ScrollBars	gördítősáv típusa többsoros szövegdoboznál
		TextAlign	a szöveg igazítása
		WordWrap	automatikus sortörés engedélyezése (többsoros szövegdoboznál)
		SelectAll()	a tartalom kijelölése

Megjegyzés: ha képet rendelünk egy vezérlőelemhez (Image tulajdonság), akkor a Select Resource ablakban:
Local Resource: csak az adott vezérlőelemhez tartozik a kép.
Project Resource File: a képet felveszi a projekt erőforrásai közé.

Vezérlőelemek futásidejű létrehozása, törlése

Egy vezérlőelem (beleértve az űrlapot is) Controls kollekciója (gyűjteménye, objektumokat tartalmazó „tömbje”) tartalmazza a hozzá tartozó vezérlőelemeket. A kollekció elemeit indexelhetjük (0-tól kezdve), vagy a nevükkel hivatkozhatunk rájuk.

Létrehozás, megjelenítés

1. A vezérlőelem-objektum létrehozása:
   `Dim` *`vezérlőelem-objektum`* `As` *`vezérlőelem-típus`*
   `vezérlőelem-objektum` `= New` *`vezérlőelem-típus`*
2. A tulajdonságok beállítása
3. Hozzáadás a szülőobjektum (például Form1) Controls kollekciójához:
   `szülőobjektum``.Controls.Add(`*`vezérlőelem-objektum`*`)`
   A hozzáadás következtében meg is jelenik a vezérlőelem a (látható) ablakban.
4. Eseménykezelő eljárás hozzárendelése a vezérlőelemhez:
   `AddHandler` *`vezérlőelem-objektum.esemény`*`, AddressOf` *`eseménykezelő eljárás neve`*

Megjegyzés: ha ciklussal hozzuk létre a vezérlőelemeket, akkor ügyeljünk arra, hogy a New operátor a cikluson belül helyezkedjen el! Az eseménykezelő eljárásban a **sender** paraméter adja meg az eseményhez kapcsolódó vezérlőelem-objektumot.

Törlés

1. Az eseménykezelő eljárás hozzárendelésének megszüntetése:
   `RemoveHandler` *`vezérlőelem-objektum.esemény`*`, AddressOf` *`eseménykezelő eljárás neve`*
2. A vezérlőelem törlése a Controls kollekcióból:
   `szülőobjektum``.Controls.Remove(`*`vezérlőelem-objektum`*`)`
3. Az erőforrások felszabadítása:
   `vezérlőelem-objektum``.Dispose()`

A Controls kollekció tulajdonságai és metódusai

Névtér: System.Windows.Forms.Form
A Controls kollekció az űrlapobjektumok egy tulajdonsága. Az űrlap vezérlőelemeit tartalmazza. Ciklusokkal dolgozható fel.
Hivatkozás: *szülőobjektum*.Controls.*tulajdonság*, *szülőobjektum*.Controls.*metódus*(*argumentum*)
Argumentumok
Vezérlőelem: a változó neve, *index*: a vezérlőelem indexe a kollekcióban, *azonosító*: a vezérlőelem azonosítója **sztringként megadva**

Count	A vezérlőelemek száma a kollekcióban.
Item(*index*), Item(*azonosító*)	Hivatkozás a kollekció egy elemére index, illetve azonosító alapján. az Item kulcsszó elhagyható: Controls(*index*), Controls(*azonosító*)
Add(*vezérlőelem*)	Hozzáadja a *vezérlőelem*-et a kollekcióhoz (a végére kerül).
Contains(*vezérlőelem*), ContainsKey(*azonosító*)	Tartalmazza-e a *vezérlőelem*-et (True/False).
IndexOf(*vezérlőelem*), IndexOf(*azonosító*)	Megadja a vezérlőelem indexét (–1, ha nincs benne a kollekcióban).
Remove(*vezérlőelem*), RemoveAt(*index*), RemoveByKey(*azonosító*)	Törli a megadott vezérlőelemet a kollekcióból.

Események

	Esemény	Az eseményobjektum legfontosabb tulajdonságai
Az ablak betöltése (megjelenítése)	Load	
Az ablak bezárásának kezdeményezése	FormClosing	Megszakítható az e.Cancel = True beállításával. CloseReason: a bezárás oka (CloseReason típusú felsorolás)
Az ablak bezárásának befejezése	FormClosed	CloseReason: a bezárás oka (CloseReason típusú felsorolás)
Az ablak méretének megváltozása	SizeChanged	A Size tulajdonság módosításakor következik be.
Billentyű lenyomása és felengedése (betű, szám, írásjel, Ctrl, Enter)	KeyPress	KeyChar: a beírt karakter (karakter típusú, módosítható!) a–z, A–Z, 0–9, írásjelek, ChrW(Keys.Control), ChrW(Keys.Enter)
Billentyű lenyomása, felengedése	KeyDown, KeyUp	Alt, Control, Shift: lenyomták-e a vezérlőbillentyűt KeyCode: a lenyomott billentyű Keys-kódja
Kattintás az egérrel vagy Enter a fókuszban lévő parancsgombon	Click, MouseClick	Button: melyik egérgombbal történt a kattintás (MouseButtons típusú felsorolás) Clicks: a kattintások száma Delta: +120 a görgő előre-, –120 a görgő hátrafelé történő forgatásánál X, Y: a kattintás pozíciója Az egéresemények sorrendje kattintáskor: MouseDown, Click, MouseClick, MouseUp, MouseDown, DoubleClick, MouseDoubleClick, MouseUp.
Dupla kattintás az egérrel	DoubleClick, MouseDoubleClick	
Egérgomb lenyomása, felengedése	MouseDown, MouseUp	
Egérgörgő mozgatása	MouseWheel	

Az egér belép egy vezérlőelem területére	MouseEnter	Az egéresemények sorrendje mozgatáskor: MouseEnter, MouseMove, MouseHover, MouseLeave.
Az egér mozgatása	MouseMove	
Az egér egy vezérlőelem területén tartózkodik	MouseHover	
Az egér elhagyja egy vezérlőelem területét	MouseLeave	
Megváltozik a szövegdoboz tartalma	TextChanged, MultiLineChanged	

Megjegyzés: a legtöbb eseményobjektum rendelkezik a Handled tulajdonsággal, melynek True-ra állítása törli az eseményt.

A MessageBox objektumosztály

Névtér: System.Windows.Forms.
Üzenetablak megjelenítéséhez a MessageBox.Show osztálymetódust használjuk:
`MessageBox.Show(`*`üzenet`*`[, `*`címsor`*`[, `*`gombok`*`[, `*`ikon`*`[, `*`alapgomb`*`[, `*`beállítások`*`]]]]])`
Az argumentumok jelentése:

- *üzenet*: az ablakban megjelenő üzenet szövege
- *címsor*: a címsor szövege
- *gombok*: az ablakban megjelenő gombok, MessageBoxButtons típusként megadva
- *ikon*: az ablakban megjelenő ikon, MessageBoxIcon típuként megadva
- *alapgomb*: alapértelmezett gomb, az Enter lenyomása a rákattintással egyenértékű, MessageBoxDefaultButton típusként megadva
- *beállítások*: további beállítások, MessageBoxOptions típusként megadva

A függvény visszatérési értéke DialogResult típusú felsorolás (a ToString metódussal a feliratot angol nyelvű sztringként kapjuk meg).
A MessageBox argumentumainál előforduló felsorolás típusok:

MesageBoxButtons felsorolás: OK, OKCancel, AbortRetryIgnore, YesNoCancel, YesNo, RetryCancel
MessageBoxIcon: None, Hand, Question, Exclamation, Asterisk, Stop, Error, Warning, Information
MessageBoxDefaultButton: Button1, Button2, Button3
DialogResult: a kiválasztott parancsgombot jelző felsorolás (None, OK, Cancel, Abort, Retry, Ignore, Yes, No)

Üzenetet az MsgBox függvénnyel is megjeleníthetünk a képernyőn (névtér: Microsoft.VisualBasic):
`MsgBox(`*`üzenet`*`[, [`*`stílus`*`][, `*`címsor`*`]])`
A *stílus* a megjelenő gombokat és ikont határozza meg. A stílust az MsgBoxStyle felsorolás elemeinek összegeként állíthatjuk elő:
OKOnly (0), OKCancel (1), AbortRetryIgnore (2), YesNoCancel (3), YesNo (4), RetryCancel (5),
Critical (16), Question (32), Exclamation (48), Information (64), DefaultButton1 (0), DefaultButton2 (256), DefaultButton3 (512)
A függvény visszatérési értéke MsgBoxResult felsorolás-típusú:
OK (1), Cancel (2), Abort (3), Retry (4), Ignore (5), Yes (6), No (7)

A System.Windows.Forms névtér Timer objektumosztálya

Névtér: System.Windows.Forms
A konsturktor meghívása: New Timer(). Az eszköztárból is elhelyezhető az űrlapon.
Hivatkozás: *változónév*.*tulajdonságnév*, *változónév*.*metódusnév*()

Interval	A Tick események gyakorisága ezredmásodpercben
Tag	Tetszőleges sztring tárolására szolgáló tulajdonság.
Start()	Elindítja a timert (a Tick események létrehozását).
Stop()	Leállítja a timert (a Tick események létrehozását).

A program(szál) futását a System.Threading.Thread.Sleep(*várakozás*) metódussal is felfüggeszthetjük, ahol az argumentum a várakozás idejét adja meg ezredmásodpercben.
Megjegyzés: konzolalkalmazásban a System.Windows.Forms.Timer helyett használjuk a System.Threading.Timer osztály objektumait!

Fájlkezelés

Az osztályok és objektumok használatánál ügyeljünk a hozzáférési jogokra!

Szövegfájlok kezelése

Az aktuális mappa és a felhasználó Dokumentumok mappája

A nem mentett projektnél az aktuális mappa a c:\Documents and Settings*felhasználónév*\Local Settings\Application Data\Temporary Projects*projektnév*\bin\Debug mappa.
A mentett projektnél hibakereső üzemmódban az aktuális mappa a projekt mappájában a bin\Debug mappa.
Az aktuális felhasználó Dokumentumok mappája: "My.Computer.FileSystem.SpecialDirectories.MyDocuments"

Olvasás szövegfájlból

1. IO.Streamreader típusú változó deklarálása:
   `Dim változónév As IO.StreamReader`
2. IO.Streamreader objektum létrehozása:
   `változónév = New IO.StreamReader(elérésiút[, kódolás])`
   elérésiút: a fájl relatív vagy abszolút elérési útja sztringként megadva
   kódolás: a System.Text.Encoding osztály osztálytulajdonsága, például: System.Text.Encoding.ASCII, Default, Unicode, UTF8, WindowsCodePage
3. A fájl olvasása metódusokkal
4. A fájl lezárása:
   `változónév.Close()`
5. Az objektumváltozó felszabadítása:
   `változónév.Dispose()`

Az IO.StreamReader típusú objektum tulajdonságai és metódusai

Névtér: System.IO
A konstruktor meghívása: New StreamReader(*elérésiút*[, *kódolás*])
Az argumentumok jelentését lásd fent.

EndOfStream	Elértük-e a stream végét (True/False).
Close()	Lezárja a stream-et.
Dispose()	Felszabadítja a stream-hez kapcsolódó erőforrásokat.
Peek()	Beolvassa a következő karakter kódját, de nem módosítja az aktuális pozíciót. Visszatérési értéke –1, ha elértük a stream végét.
Read()	Beolvassa a következő karaktert, és megadja a kódját. Visszatérési értéke –1, ha elértük a stream végét.
Read(*karaktertömb*, *kezdőindex*, *darab*)	Beolvas legfeljebb *darab* számú karaktert, melyet a *kezdőindex*-től kezdve beír a *karaktertömb*-be. Visszatérési értéke a beolvasott karakterek száma (0 ≤ érték ≤ *darab*).
ReadLine()	Beolvas egy sort a stream-ből. Visszatérési értéke Nothing, ha elértük a stream végét.
ReadToEnd()	Beolvassa a stream hátralévő részét. Visszatérési értéke üres sztring (""), ha elértük a stream végét.

Megjegyzés: egy változó Nothing értékét az Is Nothing, illetve IsNot Nothing kifejezésekkel ellenőrizhetjük (True/False).
A fájl tartalmát előolvasás nélkül a következő ciklussal olvashatjuk be (az automatikus típuskonverzió miatt a –1 a logikai True értékének felel meg):

```
Do While Not fájlobjektum.Peek() … Loop
```

A szövegfájlok egy sorában tárolt több értéket a sztring Split metódusával választhatjuk szét (lásd ott).

Szövegfájlok létrehozása és írása

1. IO.StreamWriter típusú változó deklarálása:
   ```
   Dim változónév As IO.StreamWriter
   ```
2. IO.StreamWriter típusú objektum létrehozása:
   ```
   változónév = New IO.StreamWriter(elérésiút[, hozzáfűz][, kódolás])
   ```
   elérésiút: a fájl relatív vagy abszolút elérési útja sztringként megadva
   hozzáfűz False: felülírja a fájlt, ha létezik. True: hozzáfűz a fájlhoz, ha létezik. Ha nem létezik a fájl, akkor mindkét esetben létrehozza. Alapértelmezés: False.
   kódolás: a System.Text.Encoding osztály osztálytulajdonsága, például: System.Text.Encoding.ASCII, Default, Unicode, UTF8, WindowsCodePage. Alapértelmezett kódolás: UTF-8
3. A fájl írása metódusokkal
4. A fájl lezárása (**elmulasztása adatvesztéssel járhat!**):
   ```
   változónév.Close()
   ```
5. Az objektumváltozó felszabadítása:
   ```
   változónév.Dispose()
   ```

A StreamWriter típusú objektum metódusai

Névtér: System.IO
A konstruktor meghívása: New StreamWriter(*elérésiút*[, *hozzáfűz*][, *kódolás*])
Az argumentumok jelentését lásd fent.

Close()	Lezárja a stream-et.
Dispose()	Felszabadítja a stream-hez kapcsolódó erőforrásokat.
Write(*kifejezés*)	A *kifejezés* értékét (sztringként) a stream-be írja.
WriteLine([*kifejezés*])	A *kifejezés* értékét (sztringként) a soremelés kódjával kiegészítve a stream-be írja.

Az OpenFileDialog objektum tulajdonságai és metódusai

Névtér: System.Windows.Forms
A konstruktor meghívása: New OpenFileDialog()
Részletesebben lásd a tankönyv Megnyitás és mentés párbeszédablakokkal című leckéjében.

CheckFileExists	True esetén figyelmezteti a felhasználót, hogy nem létezik a fájl.
CheckPathExists	True esetén figyelmezteti a felhasználót, hogy nem létezik az elérési út.
DefaultExt	Az alapértelmezett kiterjesztés (ha a felhasználó nem adja meg).
FileName	A párbeszédablakban kiválasztott fájl elérési útja (írható/olvasható).
FileNames	A párbeszédablakban kiválasztott fájlok elérési útja (sztringtömb).
Filter	A fájltípus legördülő listájában megjelenő szűrők.
FilterIndex	A kiválasztott szűrő indexe.
InitialDirectory	A párbeszédablakban megjelenő mappa elérési útja.
Title	A párbeszédablak címsorában megjelenő szöveg.
Dispose()	Felszabadítja az erőforrásokat.
ShowDialog()	Megjeleníti a párbeszédablakot. Visszatérési értéke DialogResult típusú felsorolás, amely megadja a párbeszédablak bezárásának módját. A Mégse vagy Bezárás gombra kattintás esetén értéke: DialogResult.Cancel.

Megjegyzés: a Dispose metódus meghívása után nem érhetők el az objektum tulajdonságai! A párbeszédablak csak a fájl elérési útját olvassa be, de nem nyitja meg a fájlt!

A SaveFileDialog objektum tulajdonságai és metódusai

Névtér: System.Windows.Forms
A konstruktor meghívása: New SaveFileDialog()
Részletesebben lásd a tankönyv Megnyitás és mentés párbeszédablakokkal című leckéjében.

CheckFileExists	True esetén figyelmezteti a felhasználót, hogy nem létezik a fájl.
CheckPathExists	True esetén figyelmezteti a felhasználót, hogy nem létezik az elérési út.
DefaultExt	Az alapértelmezett kiterjesztés (ha a felhasználó nem adja meg).
FileName	A párbeszédablakban kiválasztott fájl elérési útja (írható/olvasható).
FileNames	A párbeszédablakban kiválasztott fájlok elérési útja (sztringtömb).
Filter	A fájltípus legördülő listájában megjelenő szűrők.
FilterIndex	A kiválasztott szűrő indexe.
InitialDirectory	A párbeszédablakban megjelenő mappa elérési útja.
Title	A párbeszédablak címsorában megjelenő szöveg.
Dispose()	Felszabadítja az erőforrásokat.
ShowDialog()	Megjeleníti a párbeszédablakot. Visszatérési értéke DialogResult típusú felsorolás, amely megadja a párbeszédablak bezárásának módját. A Mégse vagy Bezárás gombra kattintás esetén értéke: DialogResult.Cancel.

Megjegyzés: a Dispose metódus meghívása után nem érhetők el az objektum tulajdonságai!

A fájlrendszer objektumai

Az osztályok és objektumok használatánál ügyeljük a hozzáférési jogokra!
Lásd még a My.Computer névtér ismertetését.

A DriveInfo objektumosztály tulajdonságai

Névtér: System.IO
A konstruktor meghívása: New DriveInfo(*meghajtó*)
A *meghajtó* kijelölése például: "c:\ "

AvailableFreeSpace	A felhasználó rendelkezésére álló szabad hely bájtokban (Long).
IsReady	True, ha a meghajtó készen áll a használatra.
TotalFreeSpace	A meghajtón lévő üres terület nagysága bájtokban.
TotalSize	A meghajtó mérete bájtokban.

A Directory objektumosztály osztálymetódusai

Névtér: System.IO
Elérésiút: sztringkifejezés

Directory.CreateDirectory(*elérésiút*)	Létrehozza a megadott elérési útnak megfelelő mappát.
Directory.Delete(*elérésiút*[, *mindent*])	Törli a megadott mappát. *Mindent* = True esetén az almappákat is törli. Alapértelmezés: False.
Directory.Exists(*elérési út*)	True, ha létezik a megadott mappa, egyébként False.
Directory.GetCurrentDirectory()	Megadja az aktuális mappa elérési útját tartalmazó sztringet.
Directory.GetFiles(*elérésiút*[, *minta*[, *keresés*]])	Sztringtömbként megadja a meghatározott elérési út mappában található fájlokat. A *minta* sztring tartalmazhat helyettesítő karaktereket (?, *). A *keresés* System.IO.SearchOption típusú felsorolás (TopDirectoryOnly: almappákban nem keres, AllDirectories: almappákban is keres).
Directory.GetLogicalDrives()	Sztringként visszaadja a számítógéphez csatlakozó háttértárak főkönyvtárának elérési útját.
Directory.Move(*forrás*, *cél*)	A *cél* helyre mozgatja a *forrás* fájl/mappát.
Directory.SetCurrentDirectory(*elérésiút*)	A megadott elérési utat teszi aktuális mappává (a mappának léteznie kell!).

A File objektumosztály osztálymetódusai

Névtér: System.IO
Elérésiút, forrás, cél: sztringkifejezés

File.Copy(*forrás*, *cél*[, *felülír*])	A forrás fájlból másolással létrehozza a cél fájlt. *Felülír* = True esetén felülírja az esetlegesen már létező célt. Alapértelmezés: False.
File.Delete(*elérésiút*)	Törli a megadott fájlt.
File.Exists(*elérésiút*)	True, ha létezik a megadott fájl, egyébként False.
File.Move(*forrás*, *cél*)	Átmozgatja a forrásfájlt.

Grafika

A grafikai objektumok számos további tulajdonsággal és metódusokkal rendelkeznek. Részletesebben lásd a Súgót.

A Pen objektumosztály

Névtér: System.Drawing
Objektumait vonalak rajzolásához használjuk.
A konstuktor meghívása: New Pen(*szín*[, *méret*])
Szín: System.Drawing.Color típusú struktúra
Méret: a pixelben mért tollméret.

Color	A toll színe (írható, olvasható). Color típusú struktúra.
Width	A toll mérete pixelben.
Dispose()	Felszabadítja az objektumhoz kapcsolódó erőforrásokat.

Tollobjektumok

A tollobjektumokat kitöltött alakzatok rajzolásához használjuk.
Objektumosztályok:
- SolidBrush: egyetlen színnel tölti ki az alakzatot. Konstruktor: New SolidBrush(*szín*)
- TextureBrush: a megadott mintával tölti ki az alakzatot. Konstruktor: New TextureBrush(*minta*). A *minta* Image típusú objektum.
- LinearGradientBrush: a megadott színátmenettel tölti ki az alakzatot. Létrehozását és a színátmenet megadását lásd a Súgóban.

Névtér a SolidBrush, illetve TextureBrush osztálynál: System.Drawing, a LinearGradientBrush osztálynál: System.Drawing.Drawing2D

Color	A kitöltés színe (írható, olvasható). Color típusú struktúra.
Image	A kitöltés mintája. Image típusú objektum.
Dispose()	Felszabadítja az objektum által lefoglalt erőforrásokat.

A Graphics objektumosztály

Névtér: System.Drawing
Objektumaival kétdimenziós alakzatokat rajzolhatunk a hozzá tartozó objektum rajzfelületére (rajzlap).
Az objektumoknak nincs külön konstruktora. Létrehozásuk egy vezérlőelem Creategraphics metódusának meghívásával történik:

```
vezérlőelem.Creategraphics()
```

Szín: System.Drawing.Color típusú struktúra
Ecset: Brush típusú objektum a kitöltött alakzatok rajzolásánál.
Toll: Pen típusú objektum az alakzatok rajzolásánál.
x, y: a rajzlap x, y koordinátájú pontja, illetve az alakzatot magába foglaló téglalap bal felső csúcsának koordinátái.

Clear([*szín*])	Törli a rajzfelületet és kitölti a megadott színnel.
DrawArc(*toll*, *x*, *y*, *szélesség*, *magasság*, *kezdőszög*, *középpontiszög*)	Megrajzolja a megadott körívet. A szöget az óramutató járásával megegyező irányban mérjük, az *x* tengelytől indulva.
DrawEllipse(*toll*, *x*, *y*, *szélesség*, *magasság*)	Ellpiszist rajzol.
DrawImage(*kép*, *x*, *y*)	A megadott pozícióba (bal felső sarok) kirajzolja a megadott *kép* Image objektumot.
DrawLine(toll, x1, y1, x2, y2)	A megadott pontokat összekötő szakaszt rajzol.
DrawRectangle(*toll*, *x*, *y*, *szélesség*, *magasság*)	Téglalapot rajzol.
FillEllipse(*ecset*, *x*, *y*, *szélesség*, *magasság*)	Kitöltött ellipszist rajzol.
FillRectangle(*ecset*, *x*, *y*, *szélesség*, *magasság*)	Kitöltött téglalapot rajzol.
Graphics.FromImage(*kép*)	Az Image típusú *kép* objektumhoz létrehozza a Graphics objektumot (osztálymetódus).

A Bitmap objektumosztály

Névtér: System.Drawing
Pixelgrafika tárolására, módosítására használjuk.
A konstruktor meghívása:

New Bitmap(*képobjektum*): az Image típusú képobjektumból létrehozza a Bitmap objektumot
New Bitmap(*elérésiút*): a megadott fájlból létrehozza a Bitmap objektumot. Használható fájltípusok: BMP, GIF, EXIG, JPG, PNG, TIF
New Bitmap(*szélesség*, *magasság*): létrehozza a megadott méretű Bitmap objektumot

Grafikaobjektumot a Graphics.FromImage(*bittérkép*) osztálymetódusával hozunk létre és rendelünk hozzá a Bitmap objektumhoz.
A Bitmap objektumot az Image tulajdonság segítségével rendelhetjük hozzá egy vezérlőelemhez: *vezérlőelem*.Image = *bittérképobjektum*
Automatikus típuskonverzió hiányában használjuk a CType konverziós függvényt: *vezérlőelem*.Image = CType(*bittérképobjektum*, Image)
A rajz módosítása után célszerű meghívni a vezérlőelem Invalidate metódusát: *vezérlőelem*.Invalidate()

Height, Width	A kép mérete pixelben (a létrehozás után csak olvasható).
Dispose()	Felszabadítja az objektum által lefoglalt erőforrásokat.
GetHbitmap()	Létrehoz egy Bitmap objektumot a GDI számára, és megadja az objektum Windows-azonosítóját (handle). Az Image.FromHbitmap metódus használja fel.
GetPixel(*x*, *y*)	Megadja az *x*, *y* koordinátájú pixel színét. Color típusú struktúra.
RotateFlip(*mód*)	Elforgatja, illetve tükrözi a rajzot. A *mód* RotateFlipType típusú felsorolás, például: RotateFlipType.Rotate90FlipNone (90 fokkal forgat, nem tükröz). A további értékeket lásd az intelligens súgóban.
Save(*elérésiút*[, *típus*])	Elmenti a megadott fájlba a képet. A típust az ImageFormat osztály osztálytulajdonságaként adhatjuk meg, például: System.Drawing.Imaging.ImageFormat.Tiff. Alapértelmezett típus: PNG. A képformátum tulajdonságait (tömörítés, színmélység stb.) az *ImageCodecInfo* és az *EncoderParameters* osztály segítségével befolyásolhatjuk. Részletesebben lásd a Súgóban.

SetPixel(*x*, *y*, *szín*)	A megadott pixelt a megadott színre állítja. A *szín* Color típusú struktúra.

Megjegyzés: ha képfájlból hozzuk létre a Bitmap objektumot, akkor a fájl az objektum felszabadításáig foglalt marad (például nem menthető a módosítás). Ennek elkerüléséhez először töltsük be egy ideiglenes változóba, majd az Image.FromHBitmap metódussal készítsünk róla másolatot. Ezután az eredeti Bitmap objektumot már felszabadíthatjuk:

```
Dim Temp, Rajz As Bitmap
Temp = New Bitmap(elérésiút)
Rajz = Image.FromHbitmap(Temp.GetHbitmap()) ' átmásoljuk a Temp tartalmát
Temp.Dispose()
```

Az Image objektumosztály

Névtér: System.Drawing
A képet megjelenítő vezérlőelemek rendelkeznek Image tulajdonsággal, amely Image típusú objektum.

Height, Width	A kép mérete pixelben (írható, olvasható).
Clone()	Létrehozza a kép másolatát, amit egy Image-típusú objektumnak adhatunk át. A megjelenítéshez hívjuk meg az objektum Refresh metódusát!
Dispose()	Felszabadítja az objektum által lefoglalt erőforrásokat.
Image.FromFile(*elérésiút*)	Hozzárendeli az objektumhoz az elérésiút által meghatározott képfájlt. Osztálymetódus!
RotateFlip(*mód*)	Elforgatja, illetve tükrözi a rajzot. A *mód* RotateFlipType típusú felsorolás, például: RotateFlipType.Rotate90FlipNone (90 fokkal forgat, nem tükröz). A további értékeket lásd az intelligens súgóban.
Save(*elérésiút*[, *típus*])	Elmenti a megadott fájlba a képet. A típust az ImageFormat osztály osztálytulajdonságaként adhatjuk meg, például: System.Drawing.Imaging.ImageFormat.Tiff. Alapértelmezett típus: PNG. A képformátum tulajdonságait (tömörítés, színmélység stb.) az *ImageCodecInfo* és az *EncoderParameters* osztály segítségével befolyásolhatjuk. Részletesebben lásd a Súgóban.

A grafika frissítése

Ha egy vezérlőelem megjelenik, vagy újra megjelenik a képernyőn, akkor a program meghívja az objektum Paint eseménykezelő eljárását.

```
Sub vezérlőelem_Paint(ByVal sender As Object, ByVal e As System.Windows.Forms.PaintEventArgs)
…
End Sub
```

sender: a vezérlőelem-objektum
e: a Paint esemény tulajdonságait tartalmazó objektum. Graphics tulajdonságával érhetjük el a sender objektumhoz kapcsolódó Graphics objektumot.
A Paint eseménykezelőben készített ábrák a frissítés során újra megrajzolásra kerülnek.
A Paint eseménykezelőt a program már az indítás során meghívja!
Megjegyzés: célszerű a Paint eseménykezelőben létrehozott objektumok erőforrásait (Pen, Brush stb.) a Dispose utasítással felszabadítani.

A LINQ[6]

A LINQ (a programozási nyelvbe integrált lekérdezés) az SQL elemeivel bővíti ki a Visual Basic eszközeit.
A lekérdezések kifejezéseket alkotnak, melyeket más kifejezésekhez hasonlóan (például értékadó utasításokban) alkalmazhatunk.
A LINQ elemeit csak nagyon vázlatosan mutatjuk be. Részletesebben lásd a Visual Basic súgójában.

Egyszerű lekérdezések

Az egyszerű lekérdezés a From záradékból és további záradékokból tevődik össze:

```
From … [további záradékok]
```

(Az összesítő lekérdezéseket lásd később.)
A From-nak az első helyen kell állnia, a többi záradék sorrendje tetszőleges. A From-ot kivéve bármely záradék elhagyható.
A lekérdezés eredményét egy felsoroló (IEnumerable) objektumban tároljuk. A névtelen (anonymous) típus alkalmazásakor a program maga dönti el az eredmény típusát:

```
Dim változónév = From …
```

A lekérdezés eredményét a ToList, ToArray stb. metódusokkal a megfelelő típusú kollekcióvá alakíthatjuk, illetve ciklusokkal átalakítás nélkül is feldolgozhatjuk.
Megjegyzés: a névtelen típus használatakor a program elején helyezzük el az *Option Infer On* direktívát (vagy a *Tools*/*Options*/*Projects and Solutions*/*VB Defaults* menüben kapcsoljuk *On* állásba)!

A From záradék

A From záradék szintaxisa:

```
From változónév1 [As típus1] In adatforrás1[, változónév2 [As típus2] In adatforrás2[, …]]
```

változónév: az adatforrás egy-egy rekordját (a kollekció egy elemét) képviseli a záradékokban.
típus: a rekord/elem típusa. Ha nem adjuk meg, akkor a program az adatforrás alapján maga dönti el.
adatforrás: tömb, lista vagy bármilyen felsoroló (IEnumerable) objektum (kollekció).
Egynél több változónév/adatforrás megadása egymásba ágyazott From záradékokat jelent.

A Where záradék

A Where záradék a rekordokra vonatkozó feltételt tartalmazza.

```
Where feltétel
```

feltétel: igaz vagy hamis értékű kifejezés.
Ha a *feltétel* függvényhívást tartalmaz, akkor kiértékelésére a lekérdezés definíciójakor kerül sor (nem pedig a végrehajtásakor).

Az Order By záradék

Az Order By záradék a rekordok rendezését írja elő.

```
Order By kifejezés1 [Ascending | Descending][, kifejezés2 […]]
```

kifejezés: egy vagy több mezőnév, illetve kifejezés, egymástól vesszővel elválasztva. A rendezési sorrend a mezőnevek sorrendjének felel meg (balról jobbra).
Ascending: növekvő, Descending: csökkenő sorrend. Alapértelmezés: növekvő.

A Let záradék

A Let záradék a megadott kifejezéshez rendel változónevet (alias):

```
Let változónév = kifejezés[, …]
```

A változónév felhasználható a további záradékokban.

[6] Itt csak a LINQ in SQL-t mutatjuk be.

A Distinct záradék

A Distinct záradék elhagyja az ismétlődő értékeket a lekérdezés eredményéből:

```
Distinct
```

A Skip, Take, Skip While, Take While záradék

A záradékokkal korlátozhatjuk a lekérdezésben részt vevő rekordok számát:

```
Skip darab
Take darab
Skip While feltétel
Take While feltétel
```

darab: egész értékű kifejezés
feltétel: igaz vagy hamis értékű kifejezés
Skip: elhagyja a megadott számú elemet a kollekció elejéről. A lekérdezés csak a további elemekre vonatkozik.
Take: a lekérdezés a megadott számú elemre vonatkozik. A Skip-pel együtt a kollekció egy részének kijelölésére használhatjuk fel. A Skip megadja az első elem indexét, a Take pedig az elemek számát. Ebben az esetben a Skip-nek meg kell előznie a Take-et.
Skip While: a kollekció elejéről elhagyja azokat az elemeket, melyekre igaz a kifejezés értéke. Az első hamis értéktől kezdve a további elemek már részt vesznek a lekérdezésben (akkor is, ha rájuk is igaz a kifejezés értéke).
Take While: csak addig folytatja a lekérdezés végrehajtását, amíg igaz a kifejezés értéke. Az első hamis értéknél befejezi a lekérdezés kiértékelését (akkor is, ha van még utána olyan elem, amelyre igaz lenne a kifejezés).

A Select záradék

A Select záradék a lekérdezés eredményébe kerülő mezőket választja ki:

```
Select [változónév1 = ]mezőnév1[, [változónév2 = ]mezőnév2 […]]
```

A Select záradék választja ki az adatforrás elemeit.
A Select záradék alkalmazása nem kötelező a lekérdezésben. Hiányában a lekérdezés az adatforrás összes elemét visszadja.
A *Select* záradék után csak a *Select*-ben felsorolt mezőkre, illetve változónevekre hivatkozhatunk (az előző záradékokban definiált változók már nem érvényesek)! Nem írhatjuk ki a mezőnevek elé a minősítést! Ha változóneveket adtunk a mezőknek, akkor a mezőnevek helyett kötelező a változóneveket alkalmazni!
Egy lekérdezés több Select záradékot is tartalmazhat. Az eredményt az utolsó Select által kiválasztott elemek adják.
Ha a Select csak egyetlen mezőt vagy kifejezést tartalmaz, akkor a lekérdezés eredménye nem mezőkből álló rekordokat, hanem a kiválasztott értéknek megfelelő típusú elemeket fog tartalmazni!

Összesítő lekérdezések

Az összesítő lekérdezés az Aggregate záradékból és további záradékokból tevődik össze:

```
Aggregate … [további záradékok] …
```

Az Aggregate záradéknak az első helyen kell állnia, a többi záradék sorrendje tetszőleges. Az Aggregate kivételével bármely záradék elhagyható.
A lekérdezés eredményét egy felsoroló (IEnumerable) objektumban tároljuk: A névtelen (anonymous) típus alkalmazásakor a program maga dönti el az eredmény típusát:

```
Dim változónév = Aggregate …
```

A lekérdezés eredményét a ToList, ToArray stb. metódusokkal a megfelelő típusú kollekcióvá alakíthatjuk, illetve ciklusokkal átalakítás nélkül is feldolgozhatjuk.
Egy lekérdezésben kötelező alkalmazni a From vagy az Aggregate záradék valamelyikét.
Megjegyzés: a névtelen típus használatakor a program elején helyezzük el az *Option Infer On* direktívát (vagy a *Tools*/*Options*/*Projects and Solutions*/*VB Defaults* menüben kapcsoljuk *On* állásba).

Az Aggregate záradék

Aggregáló függvényeket alkalmaz az adatforrásokra.

```
Aggregate változónév1 [As típus1] In adatforrás1[, változónév2 [As típus2] In adatforrás2[, …]] _
  [további záradékok] _
  Into aggregálólista
```

változónév: az adatforrás egy-egy rekordját (a kollekció egy elemét) képviseli a záradékokban.
típus: a rekord/elem típusa. Ha nem adjuk meg, akkor a program az adatforrás alapján maga dönti el.
adatforrás: tömb, lista vagy bármilyen felsoroló (IEnumerable) objektum (kollekció).
további záradékok: további Aggregate, Group By záradékok, illetve a From záradéknál bemutatott egyéb záradékok.
aggregálólista: aggregáló függvényeket tartalmazó egy vagy több kifejezés, egymástól vesszővel elválasztva.
Egynél több kifejezést tartalmazó kifejezéslista esetén a lekérdezés a kifejezéseket tartalmazó rekordot eredményez.
Egynél több változónév/adatforrás megadása egymásba ágyazott Aggregate záradékokat jelent.
Az Aggregate záradék állhat egy lekérdezés elején, de része lehet bármely más lekérdezésnek (például egy From záradéknak) is.

A Group By záradék

Csoportosítja a lekérdezés eredményének elemeit. A csoportosítás kulcsokon alapul. A csoportokra aggregáló függvényeket alkalmazhatunk.

```
Group By[ mező1[, mező2[, …]] By kulcskifejezés1[, kulcskifejezés2[, …]] Into aggregálólista
```

mező: a lekérdezés eredményében szereplő mező(k). Ha nem adjuk meg, az összes mező belekerül a lekérdezésbe.
kulcskifejezés: a csoportokat meghatározó kifejezés.
aggregálólista: az aggregálást meghatározó egy vagy több kifejezés, egymástól vesszővel elválasztva.
Az aggregálólista előtt nevet adhatunk a csoportnak: `Into változónév = Group, kifejezéslista`

Aggregáló függvények

Az adatforrás elemeiből egyetlen értéket képeznek.

All(*feltétel*)	Igaz, ha a *feltétel* minden rekordra teljesül.
Any(*feltétel*)	Igaz, ha a *feltétel* legalább egy rekordra teljesül.
Average(*mezőnév* | *kifejezés*)	A mező értékének vagy a rekordokra meghatározott kifejezések átlaga.
Count(*mezőnév* | *feltétel*)	Az összes rekord vagy a feltételnek megfelelő rekordok száma.
Group	Lásd a Visual Basic súgójában.
LongCount	Mint a Count, de visszatérési értéke Long típusú.
Max(*mezőnév* | *kifejezés*)	A mezőértékek vagy a rekordokra meghatározott kifejezések maximuma.
Min(*mezőnév* | *kifejezés*)	A mezőértékek vagy a rekordokra meghatározott kifejezések minimuma.
Sum(*mezőnév* | *kifejezés*)	A mezőértékek vagy a rekordokra meghatározott kifejezések értékének összege.

Az aggregáló függvények a Count kivételével nem veszik figyelembe az üres mezőket.
Megjegyzés: saját aggregáló függvényeket is definiálhatunk. Részletesebben lásd a Visual Basic súgójában.

A lekérdezések definíciója és végrehajtása

A lekérdezést tartalmazó *változónév* `= {From | Aggregate}` … értékadó utasítás csak definiálja a lekérdezést. A végrehajtásra a változóra való hivatkozáskor (például kifejezésben való felhasználásakor) kerül sor. A lekérdezés azonban azonnal végrehajtásra kerül, ha

- kifejezésben szerepel (értékadó utasítás helyett);
- közvetlenül egy másik adattípusra konvertáljuk az eredményt;
- a Where záradék feltételében függvényhívás szerepel.

Office-alkalmazások osztálykönyvtárainak felvétele

A programfejlesztés előtt fel kell venni az Office-alkalmazás osztálykönyvtárát a projektbe:
MS Office 2003 esetén:
– Microsoft Word 11.0 Object Library
– Microsoft Excel 11.0 Object Library
MS Office 2007 esetén:
– Microsoft Word 12.0 Object Library
– Microsoft Excel 12.0 Object Library
Az osztálykönyvtár felvétele:
1. Project/*projektnév* Properties, References panel Add gomb, COM[7] panelen a megfelelő komponens kiválasztása, OK
2. A projekt tulajdonságlapján az Imported Namespaces listájában a megfelelő alkalmazás kijelölése

Egyéb elemek

Véletlenszám-generálás

A Random objektumosztály

Névtér: System
Egyenletes eloszlású véletlenszámokat generál.
A konstruktor meghívása:
– véletlenszerű kezdőértékkel: `New Random()`
– ismétlődő kezdőértékkel: `New Random(`*x*`)`
a kezdőérték az *x* egész számtól függ (azonos *x* esetén azonos sorozat jön létre)
Megjegyzés: a konstruktor argumentum nélküli meghívása a rendszeridőből kiindulva képezi az egyenletes eloszlású véletlenszám-sorozatot. Az óra véges pontossága miatt a szorosan egymás után létrehozott random objektumok ugyanazt a véletlenszám-sorozatot eredményezik. Több véletlenszám-objektum helyett használjunk csak egyet!

[7] Component Object Model

A Random objektum metódusai

Ügyeljünk a balról zárt, jobbról nyílt intervallumokra!

`Next()`	0 ≤ véletlenszám < Integer.MaxValue (értéke Integer)
`Next(`*`max`*`)`	0 ≤ véletlenszám < *max* (értéke Integer)
`Next(`*`min, max`*`)`	*min* ≤ véletlenszám < *max* (értéke Integer)
`NextDouble()`	0 ≤ véletlenszám < 1 (értéke Double)

A Console objektumosztály osztálytulajdonságai és osztálymetódusai

Névtér: System

Konzolalkalmazás futtatásakor a Main() eljárás kerül végrehajtásra. A parancssori argumentumokat a metódus *Argumentumok*() sztringtömbjének a segítségével érhetjük el:

```
Sub Main([ByVal Argumentumok() As String])
…
End Sub
```

Console.BackgroundColor	Háttérszín (ConsoleColor típusú felsorolás, például ConsoleColor.Blue).
Console.Beep([*frekvencia*, *időtartam*])	A megadott frekvenciájú hangot generálja a beépített hangszóróban a milliszekundumban megadott ideig.
Console.BufferHeight, BufferWidth	A képernyőpuffer maximális mérete.
Console.CursorLeft, CursorTop	Kurzorpozíció állítása, lekérdezése.
Console.ForegroundColor	Betűszín (ConsoleColor típusú felsorolás, például ConsoleColor.Blue).
Console.Title	A konzolablak címsorának felirata.
Console.WindowHeight, WindowWidth	A konzolablak mérete.
Console.WindowTop, WindowLeft	A konzolablak pozíciója a képernyőn (bal felső sarok).
Console.Clear()	Törli a konzolablakot (és a puffert).
Console.Read	Beolvassa a következő karaktert és megadja a kódját. –1, ha nincs több karakter.
Console.ReadKey([*elrejt*])	Vár egy karakter lenyomásáig, és megadja a hozzá tartozó ConsoleKeyInfo értékét. *Elrejt* = True esetén nem jeleníti meg a karaktert a képernyőn. Alapértelmezett érték: False.
Console.ReadLine()	Beolvas egy sort a konzolról.
Console.Write(*kifejezés*)	Kiírja a kifejezés értékét.
Console.WriteLine(*kifejezés*)	Kiírja a kifejezés értékét és sort emel.

Megjegyzés: a Read/Readline a standard input stream-ből olvas, a Write/WriteLine a standard output stream-be ír.

A DateTimePicker objektumosztály

Névtér: System.Windows.Forms
A konstruktor meghívása: New DateTimePicker()
A felhasználó számára megkönnyíti a dátum megadását. A dátum kiválasztása a ValueChanged esemény bekövetkezését okozza.
Az alábbiakon kívül még számos tulajdonság és metódus teszi lehetővé a vezérlőelem rugalmas felhasználását (például formázását). Részletesebben lásd a Súgóban.

MaxDate, MinDate	Az elérhető legelső, illetve legutolsó dátum.
Value	A kiválasztott dátum.

A DateTimePicker objektum rendelkezik a vezérlőelemeknél feltüntetett általános tulajdonságokkal és metódusokkal is (lásd ott).
Megyjegyzés: dátumokat a *MonthCalendar* vezérlőelemmel szintén bekérhetünk. A *MonthCalendar*-ral egyszerre két hónapot jeleníthetünk meg, így hosszabb időtartam választható ki.

A Stopwatch objektumosztály

Névtér: System.Diagnostics
A konstruktor meghívása: New Stopwatch()

Elapsed	Az eltelt idő az első indítás vagy az utolsó Reset óta. A visszatérési érték TimeSpan típusú.
ElapsedMilliseconds	Az eltelt idő az első indítás vagy az utolsó Reset óta ezredmásodpercben.
ElapsedTicks	Az eltelt idő tick egységekben. A tick a stopper által mérhető legkisebb időtartam. Értéke: 1/StopWatch.Frequency másodperc.
IsRunning	Méri-e az időt a stopper (a Start és a Stop között). True, ha igen.
Reset()	Leállítja a stoppert és nullázza a mérést.
Start	Elindítja, vagy továbbfolytatja (egy Stop után) az időmérést.
Stop	Felfüggeszti az időmérést (a Start-tal folytatható).

A My.Computer névtér objektumai

Objektumok: Audio, Clipboard, Clock, FileSystem, Info, Keyboard, Mouse, Name, Network, Ports, Registry, Screen
Az objektumok néhány tulajdonsága és metódusa:

Audio.Play(*elérésiút*[*mód*])	Lejátssza a megadott .wav hangfájlt. A *mód* AudioPlayMode típusú felsorolás.
Audio.PlaySystemSound(*hang*)	Lejátssza a megadott rendszerhangot. *Hang*: System.Media.Systemsounds.Asterisk, Beep, Exclamation, Hand, Question.
Audio.Stop()	Leállítja a háttérben lejátszott hangfájlt (pl. AudioPlayMode.BackgroundLoop esetén).
FileSystem.CurrentDirectory	Az aktuális mappa elérési útja (írható/olvasható).
FileSystem.SpecialDirectories.MyDocuments	Az aktuális felhasználó Dokumentumok mappájának elérési útja.

FileSystem.RenameDirectory(*eléréisút, újnév*) FileSystem.RenameFile(*elérésiút, újnév*)	A megadott elérési útnak megfelelő mappát/fájlt átnevezi az *újnév*-re.
KeyBoard.AltKeyDown, CtrlKeyDown, ShiftKeyDown	True, ha a megadott billentyű le van nyomva.
Keyboard.CapsLock, NumLock, ScrollLock	True a CapsLock, NumLock, ScrollLock bekapcsolt állapotánál.
Keyboard.SendKeys(*sztring*[, *vár*])	Elküldi a sztringként megadott billentyűt az aktív ablaknak (mintha lenyomtuk volna a billentyűzeten). *Vár* = True esetén megvárja a sztring feldolgozását (átvételét). Alapértelemezés: True. A speciális billentyűket kapcsos zárójelbe tesszük, például: {Enter}, {Down}, {Left}, {Home} stb. Részletesebben lásd a Súgóban.
Name	A számítógép neve.
Network.IsAvailable	True, ha kapcsolódik a számítógép a hálózathoz.
Network.DownloadFile(*URL*, *célfájl* [, *felhasználónév*, *jelszó* [, *folyamatjelző*, *időhatár*, *felülír*]])	Letölti a *URL*-lel megadott fájlt a *célfájl*-ba (teljes elérési út!). Szükség esetén megadható a felhasználónév és jelszó is. *Folyamatjelző* = True esetén megjelenik egy folyamatjelző ablak. Alapértelmezés: True. Időhatár milliszekundumig vár a szerver válaszára. Felülír = True esetén felülírja a létező fájlt. A metódus nem küld http-fejlécet a szervernek, amire egyes szerverek hibaüzenettel válaszolnak.
Network.UploadFile(*forrásfájl*, *URL*[, …])	Feltölti a megadott forrásfájlt. A további paraméterek megfelelnek a DownloadFile metódus paramétereinek.

Előre definiált konstansok

vbNewLine	új sor	TriState felsorolás:	
vbTab	tabulátor	TriState.True	igaz (–1)
		TriState.False	hamis (0)
		TriStateUseDefault	alapértelmezett érték (–2)

A Color struktúra tulajdonságai és metódusai

Névtér: System.Drawing

Tulajdonságok és metódusok

Hivatkozás: *azonosító.tulajdonságnév*, *azonosító.metódusnév*(*argumentumok*)

R, G, B	A szín R, G, B értéke.
A	A szín átlátszóság-értéke (alfa-komponens).
Name	A szín angol elnevezése (előre definiált színekre).

Néhány előre definiált szín megnevezése (megosztott tulajdonságok):

Color.Black	fekete	Color.Magenta	bíbor
Color.Blue	kék	Color.Red	vörös
Color.Cyan	türkizkék	Color.White	fehér
Color.Green	zöld	Color.Yellow	sárga

A többi szín kódját Color Members címszó alatt lásd a Visual Basic súgójában.

Sztatikus metódusok

Hivatkozás: Color.*metódusnév*(*argumentumok*)

FromArgb([*A*,] *R*, *G*, *B*)	Az A, R, G, B kódú szín.
FromName(*sztringkifejezés*)	A sztringként megadott színből képezett szín (előre definiált színekre).

A Keys felsorolás elemei (billentyűzetkódok)

Névtér: System.Windows.Forms

Keys.A–Keys.Z	betűk	Keys.Home	Home	Keys.Pause	Pause
Keys.Alt	Alt	Keys.LControlKey	bal oldali Ctrl	Keys.RControlKey	jobb oldali Ctrl
Keys.Back	Backspace	Keys.Left	←	Keys.Return	Return (Enter)
Keys.Control	Ctrl	Keys.LMenu	bal oldali Alt	Keys.Right	→
Keys.D0–Keys.D9	számjegyek	Keys.LShiftKey	bal oldali Shift	Keys.RMenu	jobb oldali Alt
Keys.Delete	Delete	Keys.LWin	Windows	Keys.RShiftKey	jobb oldali Shift
Keys.Down	↓	Keys.Menu	Alt	Keys.Shift	Shift
Keys.End	End	Keys.NumPad0–Keys.NumPad9		Keys.Space	szóköz
Keys.Enter	Enter		numerikus 0–9	Keys.Tab	tabulátor
Keys.Escape	Esc	Keys.PageDown	PageDown	Keys.Up	↑
Keys.F1–Keys.F12	F1–F12	Keys.PageUp	PageUp		

Tartalomjegyzék