Juhász Tibor – Kiss Zsolt:

Programozási ismeretek
Programozási ismeretek haladóknak

(Műszaki Könyvkiadó, 2011, MK–4462-3; Műszaki Könyvkiadó, 2012, MK–4578-1)

Visual Basic 2008/2010 Express Edition
Programozási összefoglaló a tankönyvekhez

Bevezetés

A Programozási összefoglaló a *Programozási ismeretek*, illetve a *Programozási ismeretek haladóknak* tankönyv (Műszaki Könyvkiadó, 2011, 2012) kiegészítése. A tankönyvekben lehetőség szerint a programozási nyelvektől függetlenítve ismertettük a programozási tudnivalókat. Az alábbiakban az online súgó alapján (lásd lent) bemutatjuk a Visual Basic 2008/2010 nyelvi elemeit és a .NET eszközeit.

Az összefoglaló csak a tankönyvekhez kapcsolódó elemekre vonatkozik. Lefedi a tankönyvek anyagát, de a definíciókban, szintaxisban az általánosság kedvéért esetenként megemlítünk olyan fogalmakat is, amelyek nem szerepelnek a könyvekben. Az azonos célra használható nyelvi elemek közül előnyben részesítjük 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 elemeket. A részletesebb áttekintést 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/Visual Basic Reference/Visual Basic Language Reference

Az objektumosztályok ismertetése: .NET Development/.NET Framework/.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 2010/Visual Studio/Visual Studio Languages/Visual Basic and C#/Visual Basic/Visual Basic Reference

MSDN Library/NET Development/.NET Framework 4/.NET Framework Class Library

Hasonló módon találjuk meg az egyéb verziók leírását.

A Visual Basic 2010 dokumentációját lásd:

Microsoft Visual Basic Language Specification 10.0

Visual Basic Language Reference (Visual Studio 2010)

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 Visual Basic nyelv szintaxisának.
A {kapcsos zárójelben} lévő, | függőleges vonallal elválasztott opciók közül az egyiket kötelező alkalmazni. A kapcsos zárójel és a függőleges vonal nem része a nyelv szintaxisának.
A Boolean típusú függvények/metódusok ismertetésénél azt adtuk meg, hogy mikor ad vissza True értéket. Ellenkező esetben a visszatérési érték értelemszerűen False.

Névterek a .NET-ben

A .NET több ezer definíciót magában foglaló osztálykönyvtára hierarchikus rendszert alkot. A rendszer csomópontjait névtérnek nevezzük. Egy névtér többek között tartalmazhat további névtereket, osztálydefiníciókat vagy struktúrákat. A névterek rendszerét egy háttértár mappaszerkezetéhez hasonlóan képzelhetjük el. A névterek lehetővé teszik az osztályok csoportosítását, megkönnyítik az áttekintést és megakadályozzák az azonosítók ütközését.
A .NET-ben egy névtérben tárolt azonosítóinak közvetlen eléréséhez importálnunk kell a névteret a projektbe (Imports utasítás), de alkalmazhatunk teljesen minősített hivatkozást is, 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 (lásd a projekt tulajdonságainál, a References panelen):

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

Windows alkalmazás létrehozásánál még a következő névtereket is importálja a fejlesztőrendszer:

- System.Drawing
- System.Windows.Forms

A felsorolásban szereplő névterek 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 (szerelvények) ismertetésére sem. Tárgyalásmódunkban minden egyes projektet külön assembly valósít meg.
Az objektumosztályok közös (shared) mezőit/tulajdonságait **megosztott mezőnek/tulajdonságnak**, statikus (shared) metódusait pedig **megosztott metódusnak** nevezzük[1]. A megosztott metódusokat és megosztott tulajdonságokat az objektumosztály nevével kell minősíteni. Ugyanezeket az elnevezéseket alkalmazzuk a struktúráknál is.

[1] Didaktikai okokból eltértünk az első kötetben alkalmazott elnevezéstől.

Alapismeretek

A nyelv szintaxisa

A Visual Basic forráskódjában általában minden utasítást külön sorba írunk.
Az utasítás folytatása a következő sorban: szóköz aláhúzásjel Enter (a 2010-es változat esetenként jelölés nélkül is megengedi a folytatást a következő sorban).

```
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 saját 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. Minden kódfájl programmodulokat tartalmaz. A programmodulokat röviden moduloknak nevezzük.
A Visual Basic (program)moduljai:

– osztályok,
– struktúrák,
– szabványos (standard) modulok (további változókkal, eljárásokkal, függvényekkel),
– interfészek.

Egy kódfájlban több modult is definiálhatunk. A program végrehajtható utasításai csak modulokon belüli függvényekben és eljárásokban 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, interfész, struktúra, szabványos modul, felsorolás)	`[Class … End Class]`
	`[Interface … End Interface]`
	`[Structure … End Structure]`
	`[Module … End Module]`
	`[Enum … End Enum]`
		Osztályok, interfészek, struktúrák, szabványos modulok, felsorolások.
		A szabványos 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!
		
		A függvényeket és eljárásokat **alprogramoknak** nevezzük.

Alprogramszintű elemek (alprogramokban, azaz függvényekben és eljárásokban 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 előírások (direktívák). Értéküket beállíthatjuk a projekt tulajdonságlapján vagy a forráskódban.

`Option Explicit {On | Off}`	On: kötelező a változók deklarálása.
`Option Strict {On | Off}`	Off: engedélyezi az automatikus típuskonverziót (bővebb és szűkebb típusra is). On: szűkebb típusra nem végzi el az automatikus típuskonverziót (elkerülve ezzel az esetleges adatvesztést)! On értéke letiltja a késői kötést (futás közbeni kötést).
`Option Compare {Binary | Text}`	Binary: sztringek összehasonlítása a Unicode alapján. Text: sztringek összehasonlítása a területi beállítások alapján (nem különbözteti meg a kisbetűket a nagybetűktől).
`Option Infer {On | Off}`	On: a deklarációban nem kötelező a típust, ezt a fordítóprogram a változó legelső 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 azokra minősítés nélkül hivatkozhatunk.

Azonosítók

Az azonosítók elnevezése

A konstansok, változók, struktúrák, felsorolások, eljárások, függvények, objektumok, osztályok, interfészek stb. azonosítói

- csak Unicode betűt, számjegyet vagy aláhúzásjelet tartalmazhatnak;
- csak 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.
Megjegyzés: 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[2]

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!).

[2] Scope

Modulszintű hatókör: a teljes programmodulra kiterjed (függetlenül a deklarálás helyétől)
Modul (programmodul): osztály, interfész, struktúra, szabványos modul.
A programmodulon belül, de az alprogramokon kívül deklarált változók (mezők) 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 (lásd még a Hozzáférési módokat).
A Public (nyilvános) hozzáférésű, modulszinten deklarált változók projektszintű hatókörrel rendelkeznek (globális változók).
A projektszintű hatókörrel rendelkező azonosítókra egy másik projektben is hivatkozhatunk.
A programmodulban definiált alprogramok alapértelmezés szerint projektszintű (pontosabban Public) hatókörrel rendelkeznek.

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 metódusaiban takarás esetén a modulszintű változókra a Me minősítéssel hivatkozunk: Me.*változónév*

Hozzáférési módok[3]

A hozzáférési mód módosítja az azonosítók hatókörét. **Az alábbiakban hangsúlyozottan csak a tankönyvekhez 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 Interface … End Interface 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, interfész, szabványos 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, interfész és struktúra, továbbá struktúra mezője (változója), modul/osztályszintű változó (mező), konstans, eljárás, függvény, tulajdonság, felsorolás.
Protected hozzáférési mód: az azonosító a saját osztályán kívül az osztály utódosztályaiban érhető el.
A Protected hozzáférés csak osztályszinten alkalmazható (interfészen, szabványos modulon, struktúrán, eljáráson belül nem írható elő).
Protected hozzáférésűek lehetnek az osztályok tagjai (osztályok, intefészek, struktúrák, mezők, konstansok, felsorolások, tulajdonságok, metódusok).
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: szabványos modul, osztály, interfész és struktúra, továbbá struktúra mezője (változója), modul/osztályszintű változó (mező), konstans, eljárás, függvény, tulajdonság, felsorolás.
Hivatkozás felvétele más projektre: Project/Add Reference, a Projects panelen kiválasztjuk a megfelelő projektet, OK.

[3] Access levels

Friend hozzáférési mód: az azonosító csak a projekt kódfájljaiból érhető el[4], azaz más projektből nem hivatkozhatunk rá.

A Friend hozzáférési mód csak programmodul-, illetve projektszinten alkalmazható (alprogramon belül nem írható elő).

Friend hozzáférésű lehet: szabványos modul, osztály, interfész és sturktúra, továbbá strutkúra mezője (változója), modul/osztályszintű változó (mező), konstans, eljárás, függvény, tulajdonság, felsorolás.

A Friend és a Protected hozzáférési mód együtt is alkalmazható.

Alapértelmezett hozzáférési módok

Programelem	**Alapértelmezett hozzáférési mód**		
	Projektszinten (1)	**Modulszinten (2)**	**Eljárásszinten/blokkszinten (3)**
	deklarálva		
Konstans	–	Private, struktúrában Public (interfészben tiltott)	lokális hatókör
Változó (Dim utasítással deklarálva)			
Alprogram (eljárás, függvény)	–	Public	–
Alprogram paramétere	–	–	lokális hatókör
Szabványos modul (Module)	Friend	–	–
Osztály	Friend	Public	–
Interfész	Friend	Public	–
Struktúra	Friend	Public	–
Felsorolás (Enumeration)	Friend	Public	–
Tulajdonság (Property)	–	Public	–

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

Elemi típusok

A Visual Basicben az elemi típusok struktúrák (a sztring pedig objektum), így tulajdonságokkal és metódusokkal rendelkeznek.

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 méretük 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 használunk!**
Hexadecimális érték literálja Oktális érték literálja	`&H`*xxxx* `&O`*xxx*			*xxxx*: számjegyek a típusnak megfelelő számban. Szükség esetén alkalmazzunk típusazonosító karaktert! Az előjeles egész típusoknál az oktális és 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 és megosztott metódusok.
Hivatkozás a megosztott tagoknál: *típusnév*.*tulajdonságnév*, *típusnév*.*metódusnév*(*argumentumok*), például: Integer.MaxValue

MaxValue, MinValue	A típus legkisebb, illetve legnagyobb értéke.
Epsilon	Valós típusok tárolható legkisebb pozitív értéke.
PositiveInfinity, NegativeInfinity IsInfinity(*kifejezés*), IsPositiveInfinity(*kifejezés*), IsNegativeInfinity(*kifejezés*)	A valós MaxValue, illetve MinValue értékét túllépő műveletek eredménye, illetve annak vizsgálata.
NaN, IsNaN(*kifejezés*)	A valós típusú 0/0 eredménye, illetve vizsgálata.
ToString([*formátumkód*])	Sztringgé alakítja a numerikus értéket. A *formátumkód* értelmezését lásd a sztringeknél! Nemcsak változónévvel, hanem literállal és kifejezéssel is minősíthető, például: `4.5.ToString("F3"),  (3 + 5).ToString("F1")`

A Char típus (struktúra) megosztott metódusai

Hivatkozás: Char.*metódusnév*(*karakter*). A táblázatban a *c* karaktert jelöl.

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

Egyetlen számjegykaraktert a Val függvénnyel konvertálhatunk a karakternek megfelelő numerikus értékké (például "8" → 8).

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*), megosztott 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 (lásd a tankönyvhöz tartozó forrásfájlok között).
DateTime.DaysInMonth(*év*, *hónap*)	A *hónap* napjainak a száma a megadott *év*-ben (megosztott metódus).
DateTime.IsLeapYear(*év*)	True, ha az *év* szökőév (megosztott metódus).
DateTime.Parse(*sztring*)	Dátum/idővé alakítja a sztringet (megosztott metódus).
DateTime.TryParse(*sztring*, *változó*)	A változóba beírja a sztring dátum/idővé alakított értékét, ha lehet. Visszatérési értéke (True/False) jelzi az átalakítás sikerét (megosztott 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*)	True, ha DateTime típusúvá alakítható a kifejezés.
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*))	A *dátum*-nak megfelelően 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. Karakterek 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(karaktertömbnév[, kezdőindex, darab])
```

Ekkor a sztring értékét a karaktertömb *darab* számú eleméből jön létre a *kezdőindex*-től kezdve.
A kezdőérték megadása után a sztring már nem módosítható (a karakterei sem)! A módosítást végző műveletek valójában egy új sztringobjektumot hoznak létre a módosított tartalommal. A program átállítja a hivatkozást az új objektumra, és törli a régit. Ezért a többi objektummal ellentétben az S2 = S1 értékadás után az S2 nem az S1-re hivatkozik, hanem megkapja az S1 által tárolt sztringet.
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 sem használhatók a karakterek módosítására! Hibához vezet, ha értékadó utasítás bal oldalán helyezkednek el!
A sztring egy karaktere (a fenti módon hivatkozva rá) már Char típusú!
Megjegyzés: módosítható sztringet a StringBuilder osztály objektumai tárolnak. 1000–2000 karakternél hosszabb sztringek esetén, illetve gyakori módosításoknál célszerű StringBuilder objektumot használni (lásd az Összetett típusok fejezetben).

A sztringobjektum tulajdonságai és metódusai

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

A *hasonlítás* argumentum 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*)	True, ha a változó tartalmazza a *sztring*-et.
CopyTo(*kezdőindex*, *karaktertömb*, *célindex*, *darab*)	A *kezdőindex*-től átmásol *darab* karaktert a *karaktertömbbe* a *célindex*-től kezdve.
EndsWith(*sztring*[, *hasonlítás*])	True, ha a változó a *sztring* sztringgel végződik.

IndexOf(*sztring*[, *kezdőindex*[, *elemszám*[, *hasonlítás*]]])	Megadja a *sztring* karaktersorozat első előfordulásának pozícióját a *kezdőindex*-től kezdve (*elemszám* darab karakteren keresztül keres). A visszatérési érték –1, ha nem fordul elő benne.
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 igazít.
Remove(*kezdőidex*[, *darab*])	A *kezdőindex*-től kezdve *darab* karaktert töröl (alapértelmezett: a végéig töröl).
Replace(*sztring1*, *sztring2*)	A *sztring1* összes előfordulását lecseréli *sztring2*-re.
Split(*elválasztójel*)	A sztringet az *elválasztójel* sztringnél szétvágva elhelyezi egy sztringtömbben. Módosítja a sztringtömb méretét! Helyettesíti a tömbobjektum létrehozását és inicializálását. Az *elválasztójel*-et sztringtömbként is megadhatjuk.
StartsWith(…)	Ugyanaz, mint az EndWith, csak a sztring elejét vizsgálja.
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[5] esetén használhatjuk a *sztringváltozó=változónév.metódusnév*.ToArray (!) metódushívást, vagy karaktertömbben tárolhatjuk az eredményt.
Lásd még: Függvények/Sztringkezelő függvények

[5] A Visual Basicben a sorozatok (tömb, lista stb.) elemeire vonatkozó több metódus úgynevezett felsoroló interfészt eredményez. A felsoroló interfészt 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 megosztott metódusai

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* argumentum 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*-be. A formátum alakja (a kapcsos zárójel része a szintaxisnak): `{index[, hossz][:formátumkód]}`, melynek részei: *index*: a kiírásra kerülő érték sorszáma az argumentumlistában (0-val kezdődik a sorszámozás) *hossz*: az adott érték számára fenntartott karakterek száma pozitív érték: jobbra igazít, negatív érték: balra igazít *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 sztringtömbben. 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 megosztott metódusai

Névtér: Microsoft.VisualBasic
A Strings osztály további megosztott metódusokat 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 metódusoknál ügyeljünk az eltérő argumentumokra!

Felsorolás

Felsorolást az Enum utasítással definiálunk:

```
[hozzáférési mód] [Shadows] Enum Felsorolásnév [As Elemtípus]
    elemnév [ = érték]
    …
End Enum
```

Az Elemtípus csak valamelyik egész típus lehet.

A Visual Basic a felsorolás elemeihez alapértelmezés szerint a 0; 1; … egész értékeket rendeli. Meg is adhatjuk egy-egy elem (egész) értékét:

```
Enum HétNapja
    Hétfő : Kedd : Szerda : Csütörtök : Péntek : Szombat : Vasárnap : Hiba = -1
End Enum
```

Két különböző elemnek adhatunk azonos értéket is. Ha nem inicializálunk egy elemet, akkor értéke az első elem esetén 0 lesz, további elemek esetén pedig eggyel nagyobb, mint a közvetlenül előtte lévő elem értéke.

Felsorolást csak projekt- vagy modulszinten definiálhatunk, eljárásokban nem.

Az elem nevét a ToString metódussal kapjuk meg, például:

```
HétNapja.Péntek.ToString()
```

Az elem nevét az értékéből a CType típuskonverziós függvény adja meg, például:

```
Console.WriteLine(CType(4, HétNapja).ToString)
```

A System.Enum osztály GetValues metódusával tömböt készíthetünk a felsorolás elemeiből (a numerikus értékekből). Argumentumként adjuk meg a felsorolás típusát, amit a legkönnyebben a GetType függvénnyel kapunk meg, például:

```
Dim Napok() As Integer
Napok = System.Enum.GetValues(GetType(HétNapja))
```

Ekkor használhatjuk például a következő ciklust:

```
Dim Nap As HétNapja
For Each Nap In Napok
  Console.WriteLine(Nap)
Next
```

Az elemek elnevezéséből a GetNames metódus készít tömböt:

```
Dim NapokNeve() As String
NapokNeve = System.Enum.GetNames(GetType(HétNapja))
```

A Visual Basic számos előre definiált felsorolással rendelkezik, például:

DateInterval	a DateAdd függvény argumentuma
MsgBoxStyle	az MsgBox-ban megjelenő parancsgombok
MsgBoxResult	a felhasználó által kiválasztott parancsgomb

A teljes listát lásd:
http://msdn.microsoft.com/en-us/library/dy7yth1w.aspx

Változók és konstansok

Deklarálás, automatikus kezdőérték

Változót/konstanst csak blokkban, eljárásban/függvényben, osztályban, struktúrában, vagy szabványos modulban deklarálhatunk. A deklaráció nem helyzekedhet el az előbb felsorolt elemeken kívül, illetve interfészben sem.

		Megjegyzés
Konstans deklarálása	`[`*hozzáférési mód*`] [Shadows]` `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	`[`*hozzáférési mód*`]` `[[Shared] [Shadows]	[Static]] [ReadOnly]]` `Dim …`  `Dim` *név*`, … As` *típus*`, …` `Dim` *név* `As` *típus* `[= kezdőérték], …` `Dim` *név* `As New` *konstruktor*`[(`*argumentumok*`)]`	Kezdőérték megadása esetén a változónév után mindig ki kell írni a típust. A *kezdőérték* fordítási időben kiértékelhető kifejezés is lehet. Hozzáférési mód, Shared, Shadows, Static megjelölés esetén elhagyható a Dim kulcsszó.
Automatikus kezdőérték (lásd még a Törölhető változók ismertetését)	Numerikus változók: 0 Char típus: bináris 0 Hivatkozás típusok: Nothing Boolean: False Date: #01/01/0001 12:00:00 AM#	objektumok, sztringek, tömbök stb.

A típust a változónév mögé írt típusazonosító karakterekkel is megadhatjuk, például `Dim X As Integer` helyett: `Dim X%`
Típusazonosító karakterek:

Integer:	%	Single:	!	Decimal:	@
Long:	&	Double:	#	String:	$

Megjegyzések:

1. Numerikus literálok esetén a forráskódban tizedespontot, az adatbevitelnél a területi beállításoknak megfelelő elválasztójelet (például tizedesvesszőt) alkalmazunk.
2. A deklarációban megadhatjuk a struktúra, illetve objektumosztály adattagjainak kezdőértékét (lásd ott).

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 ér véget az élettartamuk).
Modulszintű változók élettartama
Szabványos 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, illetve struktúrák Shared mezőit/tulajdonságait megosztott mezőnek/tulajdonságnak nevezzük.
Static: meghosszabbítja az eljárás-, illetve blokkszintű változók élettartamát. A statikus változók élettartama:

Az alprogram helye	**A statikus változó élettartamának**	
	kezdete	**vége**
Programmodul (alapértelmezés: Shared eljárás)	Az alprogram első hívása.	A program futásának befejeződése.
Osztály, struktúra (nem Shared eljárás)	Az adott osztályhoz tartozó objektum vagy struktúra típusú változó alprogramjának első hívása.	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.

Statikus változó csak eljárás-, illetve blokkszinten deklarálható, de nem szerepelhet struktúra eljárásában.
A deklarációban szerepelő kezdőértéket csak az alprogram első meghívásakor veszi fel a statikus változó.
Az objektumosztályok Shared metódusainak statikus változói az osztályhoz tartoznak, csak egyetlen példányban léteznek a memóriában. A nem Shared alprogramok statikus 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.

Törölhető (nullable) változók

A Nullable(Of *típus*) struktúra típusú változókat törölhető változóknak nevezzük. Egy törölhető változó Nothing automatikus kezdőértéket kap, illetve felveheti a Nothing értéket. Törölhető változó deklarálása:

```
Dim változónév As Nothing(Of típus)
```

vagy:

```
Dim változónév As típus?
```

vagy:

```
Dim változónév? As típus
```

A kérdőjel csak a deklarációban szerepel a változónév után, a program további utasításaiban már nem tesszük ki.
A Nothing érték lekérdezése

1. A HasValue tulajdonsággal (értéke False, ha a törölhető változó értéke Nothing, például még nem kapott értéket):
   ```
   változónév.HasValue
   ```
2. Az Is/IsNot Nothing logikai kifejezéssel, például:
   ```
   If Változónév Is Nothing Then …
   If Változónév IsNot Nothing Then …
   ```

Tömbelemek, eljárások paraméterei, függvények visszatérési értékei, objektumok tulajdonságai is lehetnek törölhető típusúak.
A hivatkozás típusú változók (tömb, sztring, osztály) nem lehetnek törölhető típusúak.

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`	A sztringek összehasonlítását lásd alább! Objektumhivatkozások összehasonlítása (Nothing-gel is!).

Az osztályoperátorokat lásd az Objektumok és objektumosztályok ismertetésénél!

Az operátorok precedenciája (elsőbbsége)

Csökkenő sorrendben:

```
^
előjel (+, –)
*, /
\
Mod
+, –
&
<<, >>
<, >, <=, >=, =, Is, IsNot, Like
Not
And, AndAlso
Or, OrElse
Xor
```

Az azonos precedenciájú műveleteket a Visual Basic balról jobbra végzi el.

Sztringek összehasonlítása

1. Relációs operátorokkal	<, >, <=, >=, =, az ábécérend szerint
2. Az StrComp függvénnyel: `StrComp(`*`sztring1`*`, `*`sztring2`*`, 1)` Függvényérték: –1 ha *sztring1* < *sztring2* 0 ha *sztring1* = *sztring2* 1 ha *sztring1* > *sztring2*	Ha az utolsó argumentum 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!
3. A Like operátorral: *`sztringkifejezés`* `Like` *`minta`*	Értéke True, ha a *sztringkifejezés* megfelel a *minta* sztringnek. A minta megfelel például az MS Access-ben használható mintáknak (*, ?, #, [*karakterlista*], [!*karakterlista*]).
4. A StringCompare megosztott metódussal	Lásd A String osztály megosztott metódusainál!

Utasítások

Az utasításokat külön sorba írjuk. A kódszerkesztő a forráskódban nem különbözteti meg egymástól a kisbetűket és a nagybetűket.
Szükség esetén folyatás a következő sorban: szóköz és aláhúzásjel a sor végére. Több utasítás egy sorban: kettősponttal elválasztva.

A legfontosabb utasítások

		Megjegyzés
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`: zárt 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 [As típus] = _`
	`  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 számláló ciklus esetén a lépésköz negatív érték.
		Exit For: kilép a ciklusból.
		Continue For: a Next-nél folytatja a ciklust.
		Option Infer On esetén a kódszerkesztő aláhúzással megjelöli, ha modulszintű változót használunk ciklusváltozóként.

		Megjegyzés
Iterátoros ciklus	`For Each` *iterátor* `[As` *típus*`] In` *kollekció* *utasítások* `Next`	Az iterátor sorra felveszi a felsoroslható kollekció elemeinek értékét. Az elemek maguk nem módosíthatók, de hivatkozás típus esetén a tagok már igen. Felsorolható kollekció például a tömb vagy a halmaz.

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()`	Egy szövegsor beolvasása a billentyűzetről.
Beolvasás Windows-alkalmazásban	Szövegdobozzal (Textbox)	Lásd: A grafikus felhasználói felület kezelése
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	Címkével (Label)	Lásd: A grafikus felhasználói felület kezelése
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. Minden kivétel csak a sorrendben első, neki megfelelő kivétel-blokk végrehajtását idézi elő. A kivételkezelés után a végül-blokk utasításai kerülnek sorra (ha van ilyen blokk), majd az End Try-t követő utasításokkal folytatódik a program végrehajtása.
A *kivétel* egy Exception vagy belőle leszármazott típusú objektumot deklarál, melynek segítségével elérhetők a kivétel tulajdonságai. A kivételtípus hiányában az adott kivétel-blokk minden kivételre vonatkozik. A kivételobjektum Message tulajdonsága megadja a kivétel angol nyelvű leírását.
When megadása esetén a kivételblokk csak akkor hajtódik végre, ha a *kifejezés* értéke True.
Exit Try esetén a végül-blokkban, ennek hiányában 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* (számkonvertálás kivétel), a *Convert* objektumosztály megosztott metódusainak a használatakor (például *Convert.ToInt32*) pedig *FormatException* (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.

Egyszerűsített kivételkezelés a fájlműveleteknél

A Using utasításblokk

A Using utasítás szerkezete:

```
Using Erőforrásnév As New Erőforrástípus
  [utasítások]
End Using
```

A fenti Using-szerkezet egyenértékű az alábbi utasításokkal

```
Dim Erőforrásnév As New Erőforrástípus
Try
  [utasítások]
Finally
  If Erőforrásnév IsNot Nothing Then
    Erőforrásnév.Dispose()
  End If
End Try
```

Például:

```
Using Fájlki As New IO.StreamWriter("ki.txt")
  Fájlki.WriteLine("Egy sor.")
End Using
```

A Using-blokk mindenképpen lezárja a fájlt és felszabadítja az erőforrásokat.

Alprogramok

Alprogramoknak tekintjük az eljárásokat és a függvényeket. Minden végrehajtható utasításnak alprogramban kell elhelyezkednie.

Eljárások

Eljárás definíciója:

```
[hozzáférési mód] [eljárásmódosítók] [Shared] [Shadows]
Sub eljárásnév([paraméterlista]) [Implements minősített metódusnevek]
  [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.
Eljárásmódosítók:

[`Overloads | Overrides | Overridable | NotOverridable | MustOverride | MustOverride Overrides | NotOverridable Overrides`]

A magyarázatot lásd az Alprogramok túlterhelése, illetve az Objektumok és objektumosztályok/Metódusok felülírása szakaszban.

Az Implements kulcsszót követő lista azoknak az eljárásoknak adja meg az interfésszel minősített nevét, melyeket implementál az eljárás. A felsorolásban vesszővel választjuk el egymástól a minősített eljárásneveket.
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 (de nem kötelezően): *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: az eseménykezelő eljárás explicit módon (egy másik eljárásból) történő meghívásánál az eseményargumentum helyére írjuk be az üres eseményt (System.EventArgs.Empty).

Függvények

Függvény definíciója:

```
[hozzáférési mód] [eljárásmódosítók] [Shared] [Shadows]
Function függvénynév[(paraméterlista)] [As visszatérési típus] [Implements miősített metódusnevek]
  [lokális deklarációk]
  [utasítások]
  [Exit Function]
  [utasítások]
End Function
```

Az utasítások között egy vagy több helyen szerepelnie kell a

```
függvénynév = kifejezés
```

értékadásnak, ami a visszatérési értéket határozza meg.
A visszatérési értéket a Return *kifejezés* utasítással is megadhatjuk, de a Return befejezi a függvény utasításainak végrehajtását. Ha nem adjuk meg a visszatérési érték típusát, akkor az Object típusú lesz.
Ha nem adunk meg visszatérési értéket, akkor az a visszatérési típus alapértelmezett értéke lesz (Object esetén Nothing).
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 eljárásmódosítók felsorolását lásd az Eljárások ismertetésénél.
Az Exit Function utasítással kiugorhatunk a függvényből.
A függvények eljárásként is hívhatók. Ebben az esetben figyelmen kívül marad a visszatérési érték.
Megjegyzés: a fordítóprogram a hatékonyabb kiértékelés érdekében átrendezheti a kifejezések operandusait. Ezért lehetőség szerint kerüljük el, hogy a függvény utasításai megváltoztassák a kifejezésben szereplő változók értékét (mellékhatás).

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 paraméterek típusának hozzáférési módja nem lehet szűkebb, mint magának az alprogramnak a hozzáférési módja (például egy Public hozzáférésű eljárás paraméterének típusa nem lehet Friend hozzáférésű struktúra).

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 lásd a súgóban (Invoke metódus)!

Alprogramok túlterhelése

Túlterhelés: egy alprogram többféle változatának definiálása ugyanazon a néven, de **egymástól eltérő** szignatúrával.
A szignatúra (prototípus) összetevői: a paraméterek száma, sorrendje és/vagy típusa.
Nem tartozik hozzá a szignatúrához a különböző módosítók megadása (például Public, Shared, Static), a paraméterek neve, a paraméterek módosítója (például ByRef vagy Optional), a függvények visszatérési értékének típusa. Nem tartozik hozzá az sem, hogy eljárásról (Sub) vagy függvényről (Function) van-e szó.
Két alprogram szignatúrája akkor különbözik, ha a hívást végző utasítás alapján meg tudjuk mondani, hogy melyik alprogramról van szó.
A túlterhelést célszerű jelölni az Overloads eljárásmódosítóval. Használata nem kötelező, de ha alkalmazzuk, akkor az összes alprogramnál ki kell írni.
Megjegyzés: szabványos modulban nem szabad kiírni az Overloads kulcsszót (de túlterhelhetünk alprogramokat).

Beépített függvények

A legfontosabb beépített függvények

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

Típus	Függvény	Megjegyzés
Sztringkezelő függvények Lásd még a sztringobjektum 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. Argumentumok *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	Argumentum: 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** (bankárkerekítés)!
Konvertálás numerikus értékre	Val(*sztringkifejezés*)	A sztring 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*)	True, ha a *kifejezés* értéke értelmezhető számként, illetve dátum/időként. Például beolvasás ellenőrzéséhez. Sztringként megadja a szám hexadecimális értékét. A numerikus kifejezés egészrésze. Elhagyja a numerikus kifejezés értékének törtrészét.

Megjegyzés: a True és False logikai érték numerikus értékre konvertálható (–1, illetve 0), így az IsNumeric(*logikai kifejezés*) értéke True.

Típuskonverziós metódusok

Névtér: System.Convert
A típuskonverziót a Convert osztály alábbi megosztott metódusaival 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 .NET típus rendelkezik a *típus*.Parse(*sztringkifejezés*) megosztott metódussal, amely a megadott típusra alakítja az argumentumsztringet.

A Math osztály tulajdonságai és metódusai

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

E	Az *e* dupla pontosságú értéke.
Pi	A π dupla pontosságú értéke.
Abs(*x*), Sqrt(*x*)	Abszolútérték, négyzetgyök.
Sin(*x*), Cos(*x*), Tan(*x*) Acos(*x*), Asin(*x*), Atan(*x*) Atan2(*y*, *x*)	Szögfüggvények (radián argumentummal!) A szögfüggvényből visszaszámolja a szöget (radiánban). Az *y*/*x* arkusz tangense (értelmezi az *x* = 0-t is).
Exp(*x*), Log(*x*) Log10(*x*)	*e* alapú hatvány, *e* alapú logaritmus 10-es alapú logaritmus
Ceiling(*x*)	A legkisebb egész, amely nagyobb vagy egyenlő az *x*-nél (felfelé kerekít).
Floor(*x*)	A legnagyobb egész, amely kisebb vagy egyenlő az *x*-nél (lefelé kerekít).
Max(*x*, *y*), Min(*x*, *y*)	A két érték maximuma/minimuma.
Pow(*x*, *y*)	x^y (hatványozás)
Round(*x*[, *n*])	*n* tizedesre kerekít (kerekítés egészre: *n* = 0).
Sign(*x*)	Előjelfüggvény (x<0 esetén –1; x = 0 esetén 0; x>0 esetén +1)
Truncate(*x*)	Elhagyja az *x* törtrészét.

Összetett típusok

Az összetett adatszerkezetek csoportosítása

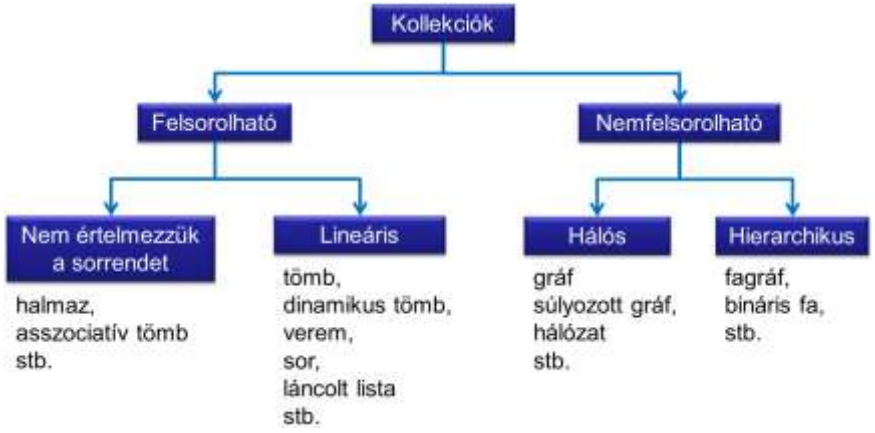
Tankönyvünkben az összetett adatszerkezeteket az alábbi csoportokba osztottuk.

A) Heterogén adattípusok: egymástól eltérő típusú adatokat is tárolhatnak
- rekord (Visual Basic: struktúra)

B) Homogén adattípusok (kollekciók vagy sokaság típusok): egyforma típusú adatokat tárolnak

1. Felsorolható típusú kollekciók
   a) Nem értelmezzük az elemek sorrendjét (nincs kapcsolat az elemek között)
      - halmaz,
      - asszociatív adattípusok (asszociatív tömb, táblázat, Visual Basic: szótár).
   b) Lineáris sorrend: lineáris kollekciók (sorozat típusok, egy–egy kapcsolat az elemek között)
      - statikus tömb,
      - dinamikus tömb (Visual Basic: lista),
      - verem,
      - sor,
      - láncolt lista, különböző változatokkal.
2. Nemfelsorolható típusú kollekciók
   a) Hálós típus: több–több kapcsolat az elemek között
      - gráf (irányított, irányítatlan gráf),
      - súlyozott gráf,
      - hálózat (speciális súlyozott gráf).
   b) Hierarchikus típus: egy–több kapcsolat az elemek között
      - fagráf (speciálisan: bináris fa).

A Visual Basic a fentiek közül a következő beépített adattípusokkal rendelkezik:
struktúra (rekord), halmaz, szótár (asszociatív tömb), statikus tömb, lista (dinamikus tömb), verem, sor, láncolt lista



Tömbök

Névtér: System
A tömbök objektumok! Hivatkozás típusú változó mutat rájuk. A *tömb2 = tömb1* értékadás után a tömb2 változó az 1. tömbre fog hivatkozni csakúgy, mint a tömb1 változó. (A program a szemétgyűjtés során 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 futtatórendszer 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 és számát jelezzük. A tömb létrehozásához használhatjuk az Array osztály CreateInstance megosztott metódusát is (részletesebbn lásd a Visual Baisc súgójában).

Hatékonyabb programot írhatunk, ha tömb helyett listát alkalmazunk. Különösen a tömbméret módosítása igényel 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 megadott *index*-től kezdve. **Többdimenziós tömb esetén is alkalmazhatjuk, sorfolytonos tárolást feltételezve.**
GetUpperBound(*i*)	Megadja az *i*-edik index legnagyobb értékét (0-val kezdődik az indexek sorszámozása).

Tömbmetódusok

Az Array osztály megosztott metódusai

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

Array.BinarySearch(*tömb*[*kezdőindex*, *darab*], *elem*[, *komparálófv*])	Bináris kereséssel megkeresi az **egydimenziós, rendezett** *tömb*-ben az *elem*-et. 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 komparáló függvény használatát lásd a Programozási ismeretek tankönyvben!
Array.Clear(*tömb*, *kezdőindex*, *elemszám*)	*elemszám* darab elem törlése a *kezdőindex*-től kezdve a *tömb*-ben. A törlés után a tömb elemeinek értéke: 0 (numerikus), False (logikai), Nothing (objektum). Eljárásként kell hívni!
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 kijelölt elem (True/False). A predikátumfüggvény értelmezését lásd a 32. 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 típusának 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 visszatérési értéke üres tömb.
Array.FindIndex(*tömb*[, *kezdőindex*[, *darab*]], AddressOf *predikátumfv*)	Ugyanaz, mint az Array.Find, de az elem indexét adja vissza. A keresést a *kezdőindex*-től 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 32. oldalon. Eljárásként kell hívni!
Array.IndexOf(*tömb*, *elem*[, *kezdőindex*[, *elemszám*])	Az *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 elemszám* darab elemének sorrendjét a *kezdőindex*-től kezdve (alapértelmezés: az egész tömb).
Array.Sort(*tömb*[, *kezdőindex*, *elemszám*] [, *komparálófv*])	Rendezi az **egydimenziós** *tömb elemszám* darab elemét a *kezdőindex*-től kezdve. A komparáló függvény értelmezését lásd a 32. oldalon, illetve a Programozási ismeretek tankönyvben!
Array.Sort(*kulcstömb*, *értéktömb*[, …])	Ugyanaz, mint az Array.Sort, de a *kulcstömb* elemeivel együtt rendezi a hozzájuk tartozó, egydimenziós *értéktömb* elemeit is.

További egydimenziós tömbmetódusok

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

All(AddressOf *predikátumfv*)	True, ha a tömb összes eleme rendelkezik-e a predikátumfüggvény által meghatározott tulajdonsággal.
Any(AddressOf *predikátumfv*)	True, ha van olyan tömbelem, amely rendelkezik a predikátumfüggvény által meghatározott tulajdonsággal.
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*])	True, ha a tömb tartalmazza a megadott értéket.
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).
First()	Visszatérési értéke a tömb első (0 indexű) eleme.
Max([AddressOf *transzformációsfv*]), Min([AddressOf *transzformációsfv*])	A (transzformációs függvény által módosított) tömbelemek maximuma, minimuma.
Last()	Visszatérési értéke a tömb utolsó (legnagyobb indexű) eleme.
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 sorozata 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 felsoroló (IEnumerable) objektum.

Megjegyzés: a Visual Basic-ben a lineáris kollekciók (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 tömbelemek feldolgozását a LINQ eszközeivel is elvégezhetjük (lásd a 69. oldalon).

Függvényparaméterek a tömbmetódusoknál

A függvényparaméterekkel rendelkező tömbmetódusok a tömb elemeit egyesével átadják a paraméterfüggvénynek, 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!

Komparáló (összehasonlító) 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*

Struktúrák (rekordok)

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

```
[hozzáférési mód] [Shadows] Structure név
[Implements interfésznevek]
  meződeklarációk
  [metódusdeklarációk]
End Structure
```

Meződeklarációk: Const, Dim, Enum utasítások
Metódusdeklarációk: Sub … End Sub, Function … End Function, Property ... End Property. A metódusok között megadhatunk konstruktort is.
A mezők és metódusok/tulajdonságok meg is oszthatók (lásd az objektumosztályoknál).
A struktúradefiníciók egymásba ágyazhatók.
A struktúra definíciójában nem adhatunk kezdőértéket a mezőknek (a megosztott tulajdonságok kivételével). Például tömb típusú mezők esetén 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 magában foglalja a konstruktor hívását:

a `Dim változónév As struktúranév` egyenértékű a `Dim változónév As struktúranév = New konstruktor()` utasítással

A struktúra-típusú változó mezőinek kezdőértékét megadhatjuk a konstruktor argumentumaként, 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.
A struktúrákban tulajdonságokat is definiálhatunk. Lásd: Az objektumok tulajdonságai.
Megjegyzés: rekordokat struktúrákban tárolhatunk. A struktúrák mezőit szokás adattagoknak is nevezni.

Halmazok

A halmaz olyan felsorolható 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)[(kollekciónév)]
```

ahol a *kollekciónév* helyére felsorolható kollekciót 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 {halmazelemek 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 metódus csökkenti.
Contains(*elem*)	True, ha a halmaz tartalmazza a megadott elemet. A futásidő független az elemszámtól!
CopyTo(*tömb*[, *index*[, *darab*]])	Átmásolja a halmaz *darab* számú elemét a *tömb*-be, a *tömb* megadott *index*-étől 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* részhalmaza a halmaznak.
Remove(*elem*)	Törli az elemet a halmazból. Visszatérési értéke True, ha megtörtént a törlés (azaz az elem szerepelt a halmazban).
SetEquals(*halmaz2*)	True, ha a halmaz és a *halmaz2* ugyanazokból az elemekből állnak. 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

A rendezett halmazban olyan sorozatot (felsorolható kollekciót) tárolhatunk, melynek nincsenek egyforma elemei. 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 felsorolható kollekciót 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 {halmazelemek 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 (vagy más lineáris kollekciók) halmazként való kezelésére. A tömbökkel halmazműveleteket végezhetünk. A műveletek végzése előtt célszerű a tömbből halmazfelsorolást készíteni, de a legtöbb halmazművelet eredménye szintén halmazfelsorolás lesz.
Megjegyzés: halmazműveleteket a LINQ eszközeivel is végezhetünk (lásd a 69. oldalon).

Halmazfelsorolás készítése

Halmazfelsorolás: olyan sorozat (felsorolható kollekció), 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.

A verem adatszerkezet

A verem „először be, utoljára ki” (LIFO) típusú lineáris kollekció.
Névtér: System.Collections.Generic
Verem deklarálása és létrehozása:

```
Dim változónév As Stack(Of típus)
változónév = New Stack(Of típus)[(kollekciónév)]
```

ahol a *kollekciónév* helyére felsorolható kollekciót vagy bármilyen, felsoroló (IEnumerable) objektumot írhatunk. Megadása esetén a program átmásolja a verembe a kollekció elemeit. A kollekciót felhasználhatjuk a verem inicializálására.
A verem tetejére a *veremnév*.Push(*kifejezés*) metódussal helyezhetünk elemet (a kifejezés értékét). A verem tetején lévő elemet a *veremnév*.Pop() utasítással vesszük ki.
A verem elemeire a feltöltés sorrendjében indexükkel is hivatkozhatunk: *változónév*.ElementAt(*index*) vagy: *változónév*(*index)*. Az indexelés a verem tetején, 0-val kezdődik.

A veremobjektum tulajdonságai és metódusai

A tulajdonságok, metódusok futásidejére vonatkozóan lásd a Visual Basic súgóját!

Count	A verem elemeinek a száma.
Clear()	Törli a verem elemeit. A memóriában fenntartott helyet a TrimExcess metódussal csökkenthetjük.
Contains(*elem*)	True, ha a verem tartalmazza a megadott elemet. Az ellenőrzést lineáris kereséssel végzi.
CopyTo(*tömb*, *index*)	Átmásolja a verem elemeit a *tömb*-be, a *tömb* megadott indexétől kezdve.
First()	Visszatérési értéke a verem első (a tetején lévő) eleme.
Last()	Visszatérési értéke a verem utolsó (az alján lévő) eleme.
Peek()	Visszaadja a verem tetején lévő elemet anélkül, hogy kivenné a veremből.
Pop()	Visszatérési értéke a verem tetején lévő elem, melyet ki is vesz a veremből.
Push(*elem*)	Elhelyezi a verem tetején az *elem* elemet.
ToArray()	Visszatérési értéke a verem elemeiből képezett tömb.
TrimExcess()	Az elemek számának megfelelő mértékben módosítja a memóriafoglalást.

A következő (vővített) metódusok ismertetése a tömböknél található: All, Any, Average, Distinct, Except, Intersect, Max, Min, Reverse, Sum, Take, TakeWhile, Union, Where.
Ezzel kapcsolatban lásd még a LINQ eszközeit (69. oldal)!

A sor adatszerkezet

A sor „először be, először ki” (FIFO) típusú lineáris kollekció.
Névtér: System.Collections.Generic
Sor deklarálása és létrehozása:

```
Dim változónév As Queue(Of típus)
változónév = New Queue(Of típus)[(kollekciónév)]
```

ahol a *kollekciónév* helyére felsorolható kollekciót vagy bármilyen, felsoroló (IEnumerable) objektumot írhatunk. Megadása esetén a program átmásolja a sorba a kollekció elemeit. A kollekciót felhasználhatjuk a sor inicializálására.
A sor végére a *sornév*.Enqueue(*kifejezés*) metódussal helyezhetünk elemet (a kifejezés értékét). A sor elején lévő elemet a *sornév*.Dequeue() utasítással vesszük ki.
A sor elemeire a feltöltés sorrendjében indexükkel is hivatkozhatunk: *változónév*.ElementAt(*index*) vagy: *változónév*(*index*). Az indexelés a sor tetején, 0-val kezdődik.

A sorobjektum tulajdonságai és metódusai

A tulajdonságok, metódusok futásidejére vonatkozóan lásd a Visual Basic súgóját!

Count	A sor elemeinek a száma.
Clear()	Törli a sor elemeit. A memóriában fenntartott helyet a TrimExcess metódussal csökkenthetjük.
Contains(*elem*)	True, ha a sor tartalmazza a megadott elemet. Az ellenőrzést lineáris kereséssel végzi.
CopyTo(*tömb*, *index*)	Átmásolja a sor elemeit a *tömb*-be, a *tömb* megadott indexétől kezdve.
Dequeue()	Visszaadja a sor elején lévő elemet, melyet ki is vesz a sorból.
Enqueue(elem)	Elhelyezi sor végén az *elem* elemet.
First()	Visszatérési értéke a sor első (az elején lévő) eleme.
Last()	Visszatérési értéke a sor utolsó (a végén lévő) eleme.
Peek()	Visszaadja a sor elején lévő elemet anélkül, hogy kivenné a sorból.
ToArray()	Visszatérési értéke a sor elemeiből képezett tömb.
TrimExcess()	Az elemek számának megfelelő mértékben módosítja a memóriafoglalást.

A következő (bővített) metódusok ismertetése a tömböknél található: All, Any, Average, Distinct, Except, Intersect, Max, Min, Reverse, Sum, Take, TakeWhile, Union, Where. Ezzel kapcsolatban lásd még a LINQ eszközeit (69. oldal)!

Listák (dinamikus tömbök)

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) a LinkedList objektumosztály valósítja meg (lásd ott).

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 {listaelemek felsorolása, vesszővel elválasztva} vagy:
változónév = New List(Of típus)(felsorolható kollekció)
```

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 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.
AddRange(*forrás*)	A lista végéhez hozzáfűzi a *forrás* elemeit. A *forrás* valamilyen felsorolható kollekció (egydimenziós tömb, lista, halmaz stb.).
Clear()	Törli a lista elemeit (Count = 0). Futásidő: egyenesen arányos az elemszámmal.
Contains(*elem*)	True, ha a lista tartalmazza a megadott elemet. Az ellenőrzést lineáris kereséssel végzi.
First()	Visszatérési értéke a lista első (0 indexű) eleme.
Insert(*index*, *elem*)	Beilleszti az *elem*-et az *index* helyre. Futásidő: egyenesen arányos az elemszámmal.
InsertRange(*index*, *forrás*)	A megadott *index*-től kezdve beilleszti a *forrás* elemeit. A *forrás* valamilyen felsorolható kollekció (egydimenziós tömb, lista, halmaz stb.).
Last()	Visszatérési értéke a lista utolsó eleme.
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.
RemoveRange(*index*, *elemszám*)	A megadott indextől kezdve törli a megadott számú elemet.
ToArray()	Visszatérési értéke a listaelemekből képezett tömb.
TrimExcess()	Az elemek számának megfelelő mértékben módosítja a memóriafoglalást.

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 felsorolható kollekciók és felsoroló (IEnumerable) objektumok a ToList() metódussal alakíthatók listává.

Láncolt lista

A Visual Basic LinkedList adattípusa kétirányú láncolt listát valósít meg, melynek elemei LinkedListNode típusú objektumok. A lista nem rendelkezik az aktuális elemet jelölő hivatkozással. Ehhez deklarálunk kell egy listaelemobjektum (LinkedListNode) típusú változót, mellyel hivatkozhatunk egy megadott listaelemre. A léptetést a Next, illetve Previous tulajdonságok segítségével tudjuk végrehajtani.

A LinkedListNode (listaelem) objektumosztály

Az objektumosztály objektumai képezik a láncolt lista (LinkedList) elemeit. Az objektumok tulajdonságai közé tartozik az elem értéke, illetve hivatkozások az előző és a következő listaelemre. Az elem értékének típusát a deklarációban kell megadni.
Névtér: System.Collections.Generic
LinkedListNode objektum deklarálása és létrehozása:

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

A listaelem értéke a létrehozáskor is megadható: `változónév = New LinkedListNode(Of típus)(érték)`

A listaelemobjektum tulajdonságai

List	Az a láncolt lista, amelyhez tartozik az objektum, egyébként pedig Nothing.
Next	A következő listaelemobjektum a listában (léptetés). Az utolsó elem esetén Nothing.
Previous	Az előző listaelemobjektum a listában (léptetés). Az első elem esetén Nothing.
Value	A listaelem értéke.

A LinkedList (láncolt lista) objektumosztály

Kétirányú listát valósít meg, melynek elemei LinkedListNode típusú objektumok. A deklarációban szereplő típus megegyezik az elemobjektumokban tárolt érték (adat) típusával.
Névtér: System.Collections.Generic
Láncolt lista deklarálása és létrehozása:

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

A lista a létrehozáskor is feltölthető elemekkel:

```
változónév = New LinkedList(Of típus)(felsorolható kollekció)
```

A láncolt lista-objektum tulajdonságai és metódusai

Count	A listaelemek száma.
First	Az első listaelem (LinkedListNode). Nothing, ha üres a lista.
Last	Az utolsó listaelem (LinkedListNode). Nothing, ha üres a lista.
AddFirst(*újlistaelem*) AddBefore(*listaelem*, *újlistaelem*) AddAfter(*listaelem*, *újlistaelem*) AddLast(*újlistaelem*)	Az első listaelem elé (First), a megadott *listaelem* elé (Before), mögé (After), illetve a lista végére (Last) beilleszti az *újlistaelem* lista-elemet. A műveletek végrehajtási ideje nem függ a lista elemszámától.
AddFirst(*érték*) AddBefore(*listaelem*, *érték*) AddAfter(*listaelem*, *érték*) AddLast(*érték*)	Az első listaelem elé (First), a megadott *listaelem* elé (Before), mögé (After), illetve a lista végére (Last) beilleszt egy új listaelemet, melynek Value tulajdonságát *érték*-re állítja. A műveletek végrehajtási ideje nem függ a lista elemszámától.
Clear()	Törli a listát (Count = 0)
Contain(*érték*)	True, ha az érték szerepel egy listaelem értékeként.
CopyTo(*tömb*, *index*)	Az *index* indextől kezdve átmásolja a *tömb* egydimenziós tömbbe a listaelemek értékét.
Find(*érték*)	Lineáris kereséssel meghatározza az első, adott *érték*-kel rendelkező listaelemet.
FindLast(*érték*)	Lineáris kereséssel meghatározza az utolsó, adott *érték*-kel rendelkező listaelemet.
Remove(*érték*)	Törli az első, adott *érték*-kel rendelkező listaelemet.. Visszatérési értéke True, ha talált ilyen listaelemet.
Remove(*listaelem*)	Törli a megadott listaelemet. Nem létező listaelem esetén kivétel jön létre.
RemoveFirst()	Törli az első listaelemet. Üres lista esetén kivétel jön létre.
RemoveLast()	Törli az utolsó listaelemet. Üres lista esetén kivétel jön létre.
ToArray()	Tömbbé alakítja a listát.

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

Asszociatív tömb (szótár)

A Visual Basic Dictionary objektumosztálya az asszociatív tömb adatszerkezetet valósítja meg. A szótárobjektum kulcs–érték párokból álló elemeket tartalmaz. A kulcs egyértelműen azonosítja a szótár elemeit (különböző elemek különböző kulcsokkal rendelkeznek). A kulcsot „indexként” használhatjuk a szótár elemeire való hivatkozásnál.

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

```
Dim változónév As Dictionary(Of kulcstípus, elemtípus)
változónév = New Dictionary(Of kulcstípus, elemtípus)
```

Hivatkozás a szótár egy elemére: *változónév*.Item(*kulcs*) vagy: *változónév*(*kulcs*).
A szótár elemei közvetlenül módosíthatók és lekérdezhetők: *érték* = *változónév*.Item(*kulcs*); *változónév*.Item(*kulcs*) = *érték*. Nem létező kulccsal rendelkező elem értékének megadása a (*kulcs, érték*) párral bővíti a szótár elemeit. Létező kulccsal rendelkező elem értékének megadása felülírja a kulcshoz tartozó értéket. Az Item hivatkozás elhagyható az utasításokból.
A szótárelemek elérési ideje független a szótár elemszámától! A tulajdonságok, metódusok futásidejére vonatkozóan lásd a Visual Basic súgóját!
Az összetett típusú kulcsokat tartalmazó szótárobjektumnál komparáló függvénnyel adhatjuk meg a kulcsok összehasonlításának módját.

A szótárobjektum tulajdonságai és metódusai

Count	A szótárban lévő kulcs–érték párok száma.
Item(*kulcs*)	A *kulcs* kulccsal rendelkező szótárelem (lásd fent).
Add(*kulcs, érték*)	Hozzáadja a kulcs–érték párt a szótárhoz. Ha már létezik a kulcs, akkor ArgumentException kivétel jön létre.
Clear()	Törli a szótár elemeit (Count = 0). Futásidő: egyenesen arányos az elemszámmal.
Contains(*kulcs*) Contains(*érték*)	True, ha a szótár tartalmazza a megadott kulccsal, illetve értékkel rendelkező elemet. Az értéket lineáris kereséssel ellenőrzi, míg a kulcs keresése független a szótár elemszámától.
Remove(*kulcs*)	Törli a megadott kulccsal rendelkező elemet a szótárból. Visszatérési értéke True, ha megtörtént a törlés, egyébként pedig False (nem tartalmazott a szótár adott kulccsal rendelkező elemet). A futásidő független az elemszámtól.

A szótárobjektum kulcsainak kollekcióját a Keys, értékeinek kollekcióját pedig a Values tulajdonság adja meg. Mindkét tulajdonság dinamikusan kapcsolódik a szótárobjektumhoz (azaz, ha változik a szótárobjektum, változnak a kollekciók). Részletesebben lásd a Visual Basic súgójában!
A következő metódusok ismertetése a tömböknél található: All, Any, Average, Distinct, Except, Intersect, Max, Min, Reverse, Sum, Take, TakeWhile, Union, Where.
A szótárobjektum a ToList() metódussal alakítható listává, illetve a ToArray() metódussal tömmbé. Mind a lista, mind a tömb KeyValuePair típusú objektumokat tartalmaz.
Egy KeyValuePair típusú elem deklarációja és létrehozása:

```
Dim változónév As KeyValuePair(Of kulcstípus, elemtípus)
változónév = New KeyValuePair(Of kulcstípus, elemtípus)
```

A KeyValuePair típusú objektumok Key tulajdonsága adja meg a kulcsot, Value tulajdonsága pedig az értéket.
A szótárobjektum elemeinek iterátora (például a For Each ciklusban) szintén KeyValuePair típusú objektum.

Rendezett kollekciók

A Visual Basic ismeri a rendezett halmaz (SortedSet), rendezett szótár (SortedDictionary), rendezett lista (SortedList) kollekciókat. A rendezett listában a szótárhoz hasonlóan kulcs-érték párokat tárolunk. (A rendezett halmaz elemeinek nincsen kulcsa.)
A rendezett kollekciók általában a rendezetlen változathoz hasonló tulajdonságokkal és metódusokkal rendelkeznek.

A StringBuilder objektumosztály

Objektumai módosítható sztringet tárolnak. 1000–2000 karakternél hosszabb sztringek esetén, illetve gyakori módosításoknál a String típus helyet célszerű StringBuilder-objektumot használni.

Névtér: System.Text
Lezárt osztály (nem definiálhatunk utódosztályokat)!
Objektum deklarálása és létrehozása:

```
Dim változónév As StringBuilder
változónév = New StringBuilder([[sztring[, kapacitás]])
```

ahol *sztring*: a változó kezdőértéke, *kapacitás*: a tárolásra fenntartott hely kezdőértéke (a karakterek száma). A kapacitás szükség esetén dinamikusan nő.
Objektumot egy sztringrésszel inicializálva is létrehozhatunk:

```
változónév = New StringBuilder(sztring, kezdőindex, hossz, kapacitás)
```

a változó kezdőértéke a megadott *sztring kezdőindex*-étől kezdve *hossz* darab karakter lesz.
Hivatkozás a sztring egy karakterére: *változónév*.Char(*index*) vagy: *változónév*(*index*). Írható/olvasható tulajdonság. Az indexelés 0-tól kezdődik.
Az objektum által tárolt sztringet a ToString metódussal érjük el (például kiírásnál).

A StringBuilder objektum tulajdonságai és metódusai

Capacity	A karakterek maximális száma (szükség esetén dinamikusan nő). A Length-nél kisebb értéket nem adhatunk meg. Maximális értéke: Integer.MaxValue.
Length	A karakterek száma. Írható/olvasható tulajdonság. Ha az aktuális hossznál kisebbre állítjuk, akkor levágja a sztring végét. Ha az aktuális hossznál nagyobbra állítjuk, akkor NULL (0 kódú) karakterekkel egészíti ki a sztringet.
Append(*kifejezés*)	Hozzáfűzi a kifejezés sztringgé konvertált értékét.
Append(*karaktertömb*[, *kezd*, *darab*])	Hozzáfűzi a *karaktertömb darab* számú elemét a *kezd* indextől kezdve. Alapértelmezés: az egész tömb.
Append(*sztring*[, *kezd*, *darab*])	Hozzáfűzi a megadott *sztring darab* számú karakterét a *kezd* indextől kezdve. Alapértelmezés: az egész sztring.
AppendFormat(*formátumsztring*, *kifejezések*)	Hozzáfűzi a kifejezés formázott értékét (a formátumsztring ismertetését lásd a String osztály megosztott metódusainál).
Clear()	Törli a sztringet.
Insert(*index*, *kifejezés*)	A megadott indexű pozíciótól kezdve beilleszti a *kifejezés* sztringgé konvertált értékét. A kifejezés karaktertömb is lehet.
Insert(*index*, *sztring*, *darab*)	A megadott indexű pozíciótól kezdve *darab* számszor beilleszti a *sztring*-et.
Remove(*kezd*, *darab*)	A *kezd* indextől kezdve töröl *darab* számú karaktert.
Replace(*sztring1*, *sztring2*[, *kezd*, *darab*])	A *kezd* indextől kezdve *darab* karakteren keresztül lecseréli a *sztring1*-et a *sztring2*-re. Alapértelmezés szerint az egész sztringben cserél.
ToString([*kezd*, *darab*])	Sztringgé konvertál a *kezd* indextől kezdve *darab* számú karaktert. Alapértelmezés: az egész sztringet megadja. Kötelező alkalmazni a sztring megjelenítésénél, illetve ha egy metódus String típusú argumentumaként adjuk meg.

Megjegyzés: az Append, Remove, Replace metódus eljárás, így az eredeti sztringet módosítja. Többször egymás után is alkalmazhatók, például:

```
változónév.Append(kifejezés1).Append(kifejezés2)…
```

Objektumok és objektumosztályok

Objektumosztály definiálása

Objektumosztályt csak projekt-, osztály-, struktúra-, szabványos modul vagy interfész szinten deklarálhatunk (azaz a deklaráció nem szerepelhet alprogramban, illetve blokkban).

A deklaráció módja

```
[hozzáférési mód] [MustInherit] [NotInheritable] Class név
  [Inherits osztálynév]
  [Implements interfésznevek]
  meződeklarációk
  tulajdonság és metódusdefiníciók
End Class
```

Módosítók:

MustInherit:	absztrakt osztály (nem példányosítható).
NotInheritable:	nem hozhatók létre utódosztályok (lezárt osztály).
Inherits:	a közvetlen ősosztály megjelölése. Nem használhatunk többszörös öröklődést.
Implements:	interfésznevek egymástól vesszővel elválasztott listája. A felsorolt interfészek összes tulajdonságát és metódusát implementálni kell.

Ha csak néhány tagot szeretnénk implementálni egy interfészből, akkor használjuk az Implements utasítást:

```
Implements interfésznév.tagnév[, interfésznév.tagnév, …]
```

Egy metódus több interfésztagot is implementálhat.

Objektumok

Objektumra hivatkozó változó deklarálása és az objektum létrehozása

```
Dim változónév As objektumosztály
változónév = New objektumosztály[(argumentumok)]
```

A deklaráció és a létrehozás összevonható egy utasításba.

A New operátor létrehozza az objektumot, majd meghívja a megadott argumentumoknak megfelelő szignatúrával rendelkező konstruktort.

A mezők és tulajdonságok kezdőértékét megadhatjuk:

1. a konstruktor argumentumaiként (az értékadást a konstruktor utasításai végzik el);
2. az objektum létrehozása után, értékadó utasításokkal (csak megfelelő hozzáférési mód alkalmazásával);
3. a New … With objektuminicializálóval (kivéve a megosztott és csak olvasható tagokat):

```
New osztály With {.tagnév = kezdőérték[, tagnév = kezdőérték …]}
```

A New … With objektuminicializálók egymásba ágyazhatók (ha a tag is objektumhivatkozás):

```
New osztály1 With {.tagnév = New osztály2 With {.tagnév = kezdőérték …} }
```

Ha nem inicializálunk egy mezőt, akkor automatikusan a típusának megfelelő kezdőértéket kapja.

Az ojbektum élettartama akkor kezdődik, amikor a New operátorral, illetve a CreateObject függvénnyel létrehozzuk, és akkor ér véget, amikor már egyetlen objektumváltozó sem hivatkozik rá.

Konstruktorok és destruktorok

Konstruktor definiálása:

```
Sub New([paraméterek])
  deklarációk
  [MyBase.New([argumentumok]) ' az ősosztály konstruktorának hívása]
  utasítások
End Sub
```

A paraméterek nélküli konstruktort alapértelmezett konstruktornak nevezzük. Létrehozása nem kötelező (lásd alább).
A konstruktorok túlterhelhetők, de nem öröklődnek.
Az utódosztály konstruktora

1. vagy legelső utasításaként, explicit módon meghívja az ősosztály valamely konstruktorát;
2. vagy implicit módon meghívja az ősosztály alapértelmezett konstruktorát.

Tehát vagy meg kell hívnunk az ősosztály valamely konstruktorát, vagy pedig léteznie kell az ősosztályban alapértelmezett konstruktornak.
Az ősosztály konstruktorának meghívása csak az utódosztály konstruktorának legelső utasítása lehet (egyéb módon nem hívható).
Destruktor definiálása:

```
Protected Overrides Sub Finalize()
  deklarációk
  utasítások
  [MyBase.Finalize() ' az ősosztály destruktorának hívása]
End Sub
```

A destruktor nem rendelkezhet paraméterekkel. Szükség esetén az ősosztály destruktorának meghívása csak az eljárás utolsó utasítása lehet.
A Visual Basic (.NET) a futtatórendszer (CIL) hatálya alá eső objektumoknál szemétgyűjtést végez. A destruktor akkor fut le, amikor a szemétgyűjtés során törli az objektumot.
A szemétgyűjtés a GC.Collect megosztott metódus hívásával is elindítható.
Egyéb objektumokat (például a fájlrendszer objektumait) a Dispose metódus hívásával szüntethetjük meg. A Dispose metódust az IDispose interfész interpretálásával saját objektumosztályainknál is definiálhatjuk. Részletesebben lásd a Visual Basic súgójában!

Tulajdonságok

Tulajdonság osztályokban, struktúrákban és szabványos modulokban definiálható (illetve interfészekben deklarálható).
Tulajdonság definiálása

```
[hozzáférési mód] [tulajdonságmódosítók] [Shared] [Shadows] [ReadOnly | WriteOnly]
Property név () [As típus] [Implements interfészek tulajdonságai]
  [hozzáférési mód] Get
    [utasítások]
  End Get
  [hozzáférési mód] Set(ByVal érték As típus)
    [utasítások]
  End Set
End Property
```

Tulajdonságmódosítók: Overloads, Overrides, Overridable, NotOverridable, MustOverride, MustOverride Overrides, NotOverridable Overrides
A Get metódus visszatérési értéke megadja a tulajdonság értékét adja meg.
A Set metódus a paramétere alapján adhat értéket a mezőknek (a tulajdonság új értéke). A tulajdonság típusának meg kell egyeznia a Set eljárás paraméterének típusával.

A tulajdonság hozzáférési módját a Property utasítással szabályozzuk. Az írható-olvasható tulajdonságok esetén vagy a Get vagy a Set metódusnál (csak az egyiknél) előírhatunk a Property-nél szűkebb hozzáférési módot.
Ha a Set eljárásban nem adunk meg paramétert, akkor a fordítóprogram implicit módon deklarál egy `value` nevű paramétert, amit felhasználhatunk az eljárás utasításaiban.
A ReadOnly tulajdonságnál csak a Get, a WriteOnly tulajdonságnál pedig csak a Set metódus definíciója szerepelhet.
A Get és a Set metódusokra a mezőnevekhez hasonlóan hivatkozhatunk, például

```
objektumnév.tulajdonságnév = kifejezés ' a Set eljárást hívja meg
változónév = objektumnév.tulajdonságnév ' a Get függvényt hívja meg
```

A Property paraméterezését, illetve az alapértelmezett tulajdonság használatát lásd a Visual Basic súgójában!
Megjegyzés: ha a forráskódban a Property utasítás utáni üres sorba g+Entert vagy s+Entert írunk, akkor a kódszerkesztő elkészíti a Get és Set metódusok vázát.

Tulajdonság automatikus implementálása

A tulajdonságok definícióját lerövidíthetjük, ha egy mezőt a következő módon deklarálunk:

```
[hozzáférési mód] Property név As típus [kezdőérték megadása]
```

A fordítóprogram automatikusan létrehozza a rejtett Get és Set metódust, melyek a tulajdonság értékét egy rejtett, Private mezőhöz rendelik hozzá.
A mező neve: *_tulajdonságnév*. Ilyen néven nem deklarálhatunk egy másik mezőt ugyanebben az osztályban.
ReadOnly és WriteOnly tulajdonságoknál nem alkalmazhatjuk az automatikus implementálást.

Megosztott mezők, megosztott metódusok

Megosztott mező: az osztályhoz tartozó mező. Értéke csak egy helyen tárolódik a memóriában, melyet közösen használnak az osztályhoz tartozó objektumok. Az osztály nevével minősítve hivatkozunk rá. A hivatkozáshoz nincs szükség objektum létrehozására.
Megosztott mező inicializálása

1. automatikus kezdőértékadással;
2. értékadással a deklarációban;
3. megosztott konstruktor (vagy megosztott metódus) hívásával;
4. objektumpéldány metódusának (például konstruktorának) a meghívásával, kihasználva a törölhető változók Nothing értékét (lásd ott).

Megosztott tulajdonságra/metódura az osztály nevével minősítve hivatkozunk. A hivatkozáshoz nincs szükség objektum létrehozására. A metódus törzsében minősítés nélkül csak megosztott mezőkre hivatkozhatunk. Nem megosztott mezőkre csak egy objektum nevével minősítve hivatkozhatunk.
A megosztást a tag deklarációjában a Shared kulcsszó jelzi (lásd a deklarációk/definíciók ismertetésénél).
Megosztott tag deklarációjában nem szerepelhet az Overrides, Overridable, NotOverridable, MustOverride és Static módosító.
A konstansok és a szabványos modulok tagjai eleve megosztott elemek, így nem írhatjuk ki a Shared módosítót.
Megjegyzés: egyes programozási nyelvekben a megosztott tagokat statikus tagoknak hívják. A Visual Basic-ben azonban a statikus megjelölés arra utal, hogy egy lokális változó az eljárásból való kilépés után is megőrzi az értékét.

Öröklődés

Az ősosztály megadása csak az osztálydefiníció első utasítása lehet (kivéve az esetleges megjegyzéssorokat):

```
Inherits ősosztálynév
```

A NotInheritable osztályokat kivéve bármely osztály szolgálhat egy másik osztály őseként. Az osztály azonban nem lehet utódja egy benne definiált, beágyazott osztálynak.
Nem lehetséges a többszörös öröklődés. Minden utódosztálynak csak egyetlen közvetlen őse lehet. Egy osztályban azonban több interfészt is implementálhatunk.
Az utódosztály hozzáférési módja nem lehet bővebb, mint az ősosztály hozzáférési módja.

Absztrakt osztályok

Absztrakt osztály: olyan osztály, amit nem példányosíthatunk. Az absztrakt osztályt a MustInherit módosítóval definiáljuk.

Lezárt osztályok

Lezárt osztály: nem lehet őse további osztályoknak.
Lezárt osztályt a NotInheritable módosítóval definiálunk. A NotInheritable nem szerepelhet együtt a MustInherit módosítóval ugyanabban a deklarációban.

A MyBase és a Myclass kulcsszó

A **MyBase** kulcsszóval a közvetlen ősosztályra hivatkozunk az utódosztályban.
A leggyakrabban akkor használjuk, ha az utódosztályban felülírtuk vagy leárnyékoltuk az ősosztály tagját. De alkalmazásával sem érhetjük el az ősosztály Private besorolású tagjait.
A MyBase nem objektumhivatkozás, hanem kulcsszó, így nem rendelhetjük hozzá egy objektumváltozóhoz, nem adhatjuk át eljárásnak stb. Nem hivatkozhat saját magára sem (például a MyBase.MyBase.*metódusnév* hibás hivatkozás).
A **MyClass** kulcsszóval az osztály aktuális példányára hivatkozunk. Abban különbözik a Me kulcsszóval történő hivatkozástól, hogy a felülírt tagok helyett is az eredeti definíciót érjük el (mintha a metódus vagy tulajdonság a NotOverridable módosítóval rendelkezett volna). Részletesebben lásd a tankönyvben!
A MyClass nem objektumhivatkozás, hanem kulcsszó, így nem rendelhetjük hozzá egy objektumváltozóhoz, nem adhatjuk át eljárásnak stb.
A MyClass nem használható Shared metódusban, de felhasználhatjuk egy példánymetódusban a Shared tagok minősítéséhez.
Megjegyzés: ha a *metódus* alprogramot az ősosztályban definiáltuk, és nem írtuk felül az aktuális osztályban, akkor a MyBase.*metódus* és a MyClass.*metódus* hivatkozások egyenértékűek.

Az Object objektumosztály

A Visual Basicben (és a .NET-ben) minden osztály közös őse az Object osztály.

Az Object osztály legfontosabb metódusai

Equals(*Objektum2*)	Visszatérési értéke True, ha az aktuális objektum hivatkozása megegyezik az Objektum2 hivatkozásával (azaz ugyanarra az objektumra hivatkoznak). Az *Objektum1*.Equals(*Objektum2*) metódushívás helyettesíthető az *Objektum1* Is *Objektum2* logikai kifejezéssel.
Finalize()	Az objektum destruktora. Lásd ott!
GetType()	Visszatérési értéke az objektum típusa.
Object.ReferenceEquals(*Objektum1*, *Objektum2*)	Megosztott metódus. Visszatérési értéke True, ha a két objektumváltozó ugyanarra az objektumra hivatkozik. Az Equals metódussal ellentétben nem írható felül. Lásd még az Osztályoperátorok szakaszhoz fűzött megjegyzést!
ToString()	Az objektum sztringgé alakításának eredménye.

Tulajdonságok és metódusok felülírása

Az ősosztály Overridable módosítóval jelölt tulajdonsága/metódusa az utódosztályban felülírható (virtuális tulajdonság/metódus).
Az utódosztályban definiált tagnak pontosan ugyanolyan névvel és szignatúrával kell rendelkeznie, mint a felülírt tagnak. A szignatúrán kívül meg kell egyeznie a hozzáférési módnak, paraméterenként a paraméterátadás típusának (ByVal vagy ByRef) és függvények esetén a visszatérési érték típusának is.
A felülírt tagra a MyBase minősítéssel hivatkozhatunk.
Ha valamely tulajdonság/metódus felülírja az ősosztály egy tagját, akkor az utódosztályokban is felülírható marad, egyébként alapértelmezés szerint nem felülírható a tag. Egy felülírt tag további felülírását a NotOverridable módosítóval tilthatjuk le az utódosztályban.
Az Overridable ugyanannál a tagnál nem szerepelhet együtt a MustOverride, NotOverridable, Overrides, Shared módosítókkal.
Az Overrides ugyanannál a tagnál nem szerepelhet együtt a Shadows vagy Shared módosítókkal.
A NotOverridable ugyanannál a tagnál nem szerepelhet együtt a MustOverride, Overridable, Shared módosítókkal.
Private tagnál nem használhatjuk az Overridable vagy a NotOverridable módosítókat.

Absztrakt metódusok

Absztrakt tulajdonság/metódus: olyan tulajdonság/metódus, melyet kötelező felülírni az utódosztályban. Az absztrakt tulajdonság/metódus nem hívható meg.
Az absztrakt tagot a MustOverride módosító jelzi a deklarációban. A deklaráción (eljárásfejen) kívül semmilyen más utasítást (például End Sub stb.) nem adhatunk meg.
Absztrakt tagot csak MustInherit (absztrakt) osztályban definiálhatunk.
A MustOverride ugyanannál a tagnál nem szerepelhet együtt a NotOverridable, Overridable, Shared módosítókkal.

Árnyékolás

Árnyékolás: egy az ősosztályban létező tag nevével megegyező tag deklarálása/definiálása az utódosztályban, a felülírás engedélyezése/megjelölése nélkül.
Bármely típusú tag bármely típusú taggal leárnyékolható (például metódus mezővel).
Az árnyékolást célszerű, de nem kötelező a Shadows módosítóval jelezni a deklarációban/definícióban. Ha az ősosztályban létező névvel deklarálunk egy tagot az utódosztályban, akkor az Overrides módosító hiányában a fordítóprogram feltételezi az árnyékolást. Erről figyelmeztető üzenetet küld az Error List ablakban.
A Shadows módosító ugyanannál a tagnál nem használható együtt az Overloads, Overrides, Static módosítóval.
Ha a leárnyékoló elem nem érhető el az utódosztályokban (például Private), akkor az utódosztályok az ősosztály leárnyékolt elemét öröklik (tehát az utódosztályokban nem érvényesül az árnyékolás).
Az árnyékolás szabályait és következményeit részletesebben lásd a Visual Basic súgójában!

Kiterjesztett metódusok

Kiterjesztett metódus: egy osztály/struktúra/interfész (így például egy tömb) metódusainak bővítése az osztálydefiníció módosítása vagy utódosztály deklarálása nélkül.
Kiterjesztett metódus csak eljárás vagy függvény lehet. Nem definiálhatunk kiterjesztett tulajdonságokat és mezőket.
A kiterjesztéshez importálni kell a System.Runtime.CompilerServices névteret.
Kiterjesztett metódust csak szabványos modulban definiálhatunk.
Kiterjesztett metódus definiálása:

```
Imports System.Runtime.CompilerServices
…
<Extension()>
[módosítók] {Sub | Function} név(paraméter[, további paraméterek]) [As típus]
  utasítások
End {Sub | Function}
```

A kiterjesztett metódus első paraméterének típusa megegyezik azzal a típussal, amit a metódus kiterjeszt. Futásidőben ezzel a paraméterrel hivatkozunk az adott típus aktuális példányára, így ezt a paramétert nem írjuk be az argumentumok közé.
A kiterjesztett metódus készítésének és alkalmazásának módját részletesebben lásd a tankönyvben, illetve a Visual Basic súgójában.

Polimorfizmus

Az objektumváltozóhoz hozzárendelhetünk olyan objektumot, amelynek típusa

1. a deklarációban megjelölt osztály valamely utódosztálya (polimorfizmus öröklődéssel);
2. olyan interfész, melynek hatókörébe tartozik a hozzárendelést végző utasítás (polimorfizmus interfésszel).

A statikus típusban nem szereplő tagokra típuskényszerítéssel vagy késői kötés alkalmazásával hivatkozhatunk (lásd az Osztályoperátorok, illetve Az Object objektumosztály ismertetését).

Osztályoperátorok

DirectCast(*kifejezés*, *típus*)	Típuskényszerítés öröklődéssel vagy interfésszel létrehozott polimorfizmus esetén. Visszatérési értéke a kifejezés, átalakítva a *kívánt típus*-ra. Nem megfelelő típus megadása InvalidCastException kivételt okoz.
objektumváltozó1 Is *objektumváltozó2*	True, ha a két objektumváltozó pontosan ugyanarra az objektumra hivatkozik.
objektumváltozó1 IsNot *objektumváltozó2*	False, ha a két objektumváltozó pontosan ugyanarra az objektumra hivatkozik.
TryCast(*kifejezés*, *típus*)	Megfelel a DirectCast operátornak, de nem megfelelő típus megadása esetén nem okoz kivételt, hanem Nothing értéket ad vissza.
TypeOf *objektumváltozó* Is *típus*	True, ha az *objektumváltozó* dinamikus típusa megfelel a megadott *típus*-nak (a *típus* osztály példánya, utódosztályának példánya vagy olyan osztály példánya, amely implementálja a *típus* interfészt). Az IsNot helyett Not Is-t kell használni.
New *típus*([*argumentumok*])	Létrehoz egy a megadott típusnak megfelelő objektumot, és meghívja az argumentumoknak megfelelő szignatúrával rendelkező konstruktort. Átadja a konstruktornak az argumentumokat. Visszatérési értéke az objektumra mutató hivatkozás.

Megjegyzés: két objektum dinamikus típusát az

```
Object.ReferenceEquals(Objektum1.GetType(), Objektum2.GetType())
```

megosztott metódushívással hasonlíthatjuk össze. Visszatérési értéke akkor True, ha a két objektum dinamikus típusa megegyezik.

Interfészek

Interfészt projektszinten, illetve programmodulokban (osztályan, interfészben, struktúrában, szabványos modulban) deklarálhatunk. Alapértelmezés szerint a projektszinten deklarált interfész Friend, a programmodulban deklarált interfész pedig Public hozzáférési móddal rendelkezik.
Az interfész definíciója csak Sub, Function, Property, Interface, Class, Sturcture és Enum utasításokat (deklarációkat) tartalmazhat.
Megengedett módosítók a tagok deklarációiban:

Shadows:	az interfész bármely tagjánál;
Overloads:	Sub, Function, Property;
ReadOnly, WriteOnly:	Property.

Egyéb módosítókat (Public, Private, Friend, Protected, Shared, Overrides, MustOverride, Overridable) nem használhatunk a deklarációkban.
A deklarált elemeket nem implementálhatjuk az interfészen belül. Az interfész tagjait osztály- vagy struktúradefiníciókban implementálhatjuk (lásd ott).
Egy interfész több interfészből is származhat.
Interfész definiálása:

```
[hozzáférési mód] [Shadows] _
Interface név
  [Inherits interfésznevek]
  [[módosítók] Property név]
  [[módosítók] Function név]
  [[módosítók] Sub név]
  [[módosítók] Interface név]
  [[módosítók] Class név]
  [[módosítók] Structure név]
End Interface
```

A tagok definícióiban nem adhatjuk meg a hozzáférési módot, mert ez az interfész minden tagjára kötelezően Public. Az implementálásnál már módosíthatjuk a hozzáférést.
A tagok között szereplő felsorolás, struktúra, osztály vagy interfész deklarációjának tartalmaznia kell adattagjaik felsorolását is.
Az interfész őseként nem jelölhetünk meg benne deklarált, beágyazott interfészt. Az ősinterfész hozzáférési módja nem lehet szűkebb az utódinterfész hozzáférési módjánál.

A grafikus felhasználói felület kezelése

Az űrlap

Az űrlap a képernyőn megjelenő programablakot és párbeszédablakokat jelképezi.
Névtér: System.Windows.Forms
A programablakra hivatkozó változó neve megegyezik a programablak osztályának nevével (Form1).

Az űrlapobjektumok legfontosabb tulajdonságai és metódusai:

AcceptButton	Az Enter lenyomása a kijelölt gombra kattintással egyenértékű.
AutoScroll	Görgetősávok automatikus megjelenítése (szükség esetén).
CancelButton	Az Esc lenyomása a kijelölt gombra való kattintással egyenértékű.
ControlBox	A rendszermenü ikonjánka megjelenítése.
Controls	Az ablak által tartalmazott vezérlőelemek kollekciója.
DialogResult	DialogResult felsorolás típusú érték. Az értékadás bezárja (elrejti) a modális párbeszédablakot. Lásd lent.
FormBorderStyle	A szegély típusa.
Icon	Az ablak ikonja.
Left, Top	Az ablak bal felső sarkának koordinátái az ablakot tartalmazó elemhez viszonyítva.
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.
Text	A címsor szövege.
WindowState	Megjelenítés módja az ablak indításánál.
Close()	Az ablak bezárása (a párbeszédablak-objektum nem szűnik meg!).
Controls.Add(*vezérlőelem-objektum*)	A vezérlőelem hozzáadása a Controls kollekcióhoz (megjelenik az ablakban). A vezérlőelemet először létre kell hozni!
Dispose()	Megszüneti a (párbeszéd)ablakobjektumot.
SetDesktopLocation(*x*, *y*)	A képernyő (*x*, *y*) koordinátájú pontjába helyezi az ablak bal felső sarkát.

A további, gyakran használatos tulajdonságokat lásd a Vezérlőelemek szakaszban.

A Controls kollekció tulajdonságai és metódusai

Névtér: System.Windows.Forms.Form
A Controls kollekció az űrlapobjektumok egy tulajdonsága, amely az űrlap vezérlőelemeit tárolja. Ciklusokkal dolgozható fel.
Hivatkozás: *szülőobjektum*.Controls.*tulajdonság*, *szülőobjektum*.Controls.*metódus*(*argumentumok*)
A táblázatban előforduló 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*)	True, ha tartalmazza a *vezérlőelem*-et.
IndexOf(*vezérlőelem*), IndexOf(*azonosító*)	Megadja a vezérlőelem indexét (–1, ha nincs benne a kollekcióban).
Remove(*vezérlőelem*), RemoveAt(*index*)	Törli a megadott vezérlőelemet a kollekcióból.

Párbeszédablakok

A párbeszédablakok a System.Windows.Forms.Form objektumosztály példányai.
Párbeszédablakobjektum létrehozása tervező nézetben: új Windows Form hozzáadása a projekthez.
Párbeszédablak megjelenítése
modális: az űrlap ShowDialog metódusával. A metódus visszatérési értéke DialogResult felsorolás típusú.
nem modális: az űrlap Show metódusával
Modális párbeszédablak megnyitásakor a hívó eljárás futása szünetel a párbeszédablak bezárásáig. A modális párbeszédablak bezárása nem szünteti meg az objektumot, csupán elrejti, így tagjaira (mezők, tulajdonságok, metódusok) továbbra is hivatkozhatunk. Ismét meg is jeleníthető, új objektumpéldány létrehozása nélkül. Ha a felhasználó az ablak Bezárás gombjára való kattintással lép ki a párbeszédablakból, akkor a DialogResult tulajdonság értéke DialogResult.Cancel lesz.
A párbeszédablak-objektum tagjaira objektumváltozó (hivatkozás) létrehozása nélkül is hivatkozhatunk. A hivatkozáshoz az osztály nevét használhatjuk (például Form1).

A DialogResult felsorolás legfontosabb elemei

A DialogResult tulajdonságnak történő értékadás bezárja a modális párbeszédablakot (de nem szünteti meg az objektumot)!

None	Nincs visszatérési érték, a párbeszédablak még nyitva van.
OK	Visszatérési érték: OK (általában az OK gombra kattintás).
Cancel	Visszatérési érték: Cancel. A felhasználó az ablak Bezárás gombjára kattintott (vagy általában a Mégse gombra kattintás).
Abort	Visszatérési érték: Abort (általában a Leállítás gombra kattintás).
Retry	Visszatérési érték: Retry (általában az Ismét gombra kattintás).

Ignore	Visszatérési érték: Ignore (általában a Kihagyás gombra kattintás).
Yes	Visszatérési érték: Yes (általában az Igen gombra kattintás).
No	Visszatérési érték: No (általában a Nem gombra kattintás).

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 tartalmazó 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,		
Height	magasság pixelben,		
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 leggyakoribb vezérlőelemek

A gyakran előforduló tulajdonságokat lásd fent! Az eseményeket a táblázatban a 💣 ikon jelöli.

Megnevezés	Osztály	Toolbox ikon	További tulajdonságok, metódusok, események	Megjegyzés
Címke	Label		TextAlign	a szöveg igazítása.
Csoportdoboz	Groupbox		Controls	a csoportdoboz által tartalmazott vezérlőelemek kollekciója.
			Text	a keret szélén megjelenő felirat.
Jelölőnégyzet	CheckBox		Checked	True, ha a jelölőnégyzetet bejelöltük, egyébként False.
			CheckedChanged 💣	esemény bekövetkezik, ha a Checked tulajdonság megváltozik.
Képdoboz	PictureBox		Image	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 méreteinek 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.
Lenyíló lista (kombinált lista)	ComboBox		Items	a lenyíló lista elemeinek kollekciója.
			MaxLength	a karakterek maximális száma.
			SelectedIndex	a kiválasztott elem indexe.
			SelectedItem	a kiválasztott elem (objektum).
			SelectedValue	a kiválasztott elem értéke.
			Sorted	True esetén az elemek rendezve jelennek meg.
			SelectedIndexChanged 💣	esemény bekövetkezik, ha megváltozik a kiválasztott elem indexe.
			SelectedValueChanged 💣	esemény bekövetkezik, ha megváltozik a kiválasztott elem értéke.
Magyarázódoboz	ToolTip		ToolTipTitle	a doboz címsorának szövege.
			SetToolTip(*vezérlőelem*, *szöveg*)	hozzárendeli a dobozt a megadott vezérlőelemhez a megadott szöveggel.

Megnevezés	Osztály	Toolbox ikon	További tulajdonságok, metódusok, események	Megjegyzés
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 karakter megjelenítése.
			ReadOnly	csak olvasható-e.
			ScrollBars	gördítősáv típusa (többsoros szövegdoboznál).
			TextAlign	a szöveg igazítása.
			WordWrap	automatikus sortörés engedélyezése (többsoros szövegdoboznál).
			SelectAll()	a tartalom kijelölése.
			TextChanged	esemény bekövetkezik, ha a Text tulajdonság értéke megváltozik.
Választógomb	RadioButton		Checked	True, ha a választógombot bejelöltük. Az azonos GroupBox-hoz tartozó választógombok közül csak egyet lehet bejelölni.
			CheckedChanged	esemény bekövetkezik, ha megváltozik a Checked tulajdonság értéke.

Megjegyzések

1. A Csoportdoboz (GroupBox) a vezérlőelemek (például választógombok) csoportosítására szolgál. A ToolBox Containers csoportjában találjuk.
2. 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.

Menüsor (Toolbox ikon:)

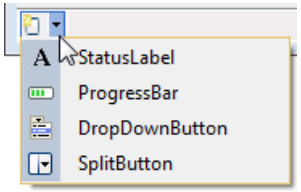
Menüsort a MenuStrip osztály objektumával készíthetünk.
Névtér: System.Windows.Forms
A menü ToolStripMenuItem típusú objektumokból áll. A menü szerkezetét (menüparancsok, almenük) tervezőnézetben hozhatjuk létre. A menüparancsokhoz billentyűkombinációt rendelhetünk, ha a parancs nevében a megfelelő karakter elé & jelet írunk.
A menü egyes parancsaihoz általában Click eseménykezelő eljárást készítünk.
Megjegyzés: a menü futásidejű létrehozásáról és módosításáról lásd a Visual Basic súgóját!
A MenuStrip osztály legfontosabb tulajdonságai:

Checked	True értéke esetén a menüparancs előtt pipakarakter (✓) jelenik meg.
Image	A menüparancs előtt megjelenő kép tulajdonságai.
Selected	True, ha a menüparancsot kiválasztották.
ShortCutKeys	A menüparancshoz tartozó billentyűparancs. Billentyűkombináció megadása például: `Keys.Control Or Keys.P`

Állapotsor

Állapotsort a StatusStrip osztály objektumával készíthetünk (Toolbox ikon:).
Névtér: System.Windows.Forms
Az állapotsor leggyakoribb elemei:

ToolStripStatusLabel szöveget (Text tulajdonság) vagy ikont (Image tulajdonság) megjelenítő címke;
ToolStripProgressBar folyamatjelző.



Egy állaposorban több elem is elhelyezhető.
Az állapotsor elemeihez általában nem rendelünk eseménykezelő eljárást.
Megjegyzés: az állapotsor futásidejű létrehozásáról és módosításáról lásd a Visual Basic súgóját!

A ToolStripProgressBar legfontosabb tulajdonságai:

Minimum, Maximum	A Value tulajdonság legkisebb, illetve legnagyobb értéke.
Size	A folyamatjelző mérete (Drawing.Size típusú struktúra).
Step	A PerformStep metódus ennyivel lépteti a folyamatjelző sávját.
Value	A kijelzett érték a Maximum és a Minimum között.
Increment(*érték*)	Az *érték*-nek megfelelő értékkel megnöveli a folyamatjelző Value tulajdonságának értékét.
PerformStep()	A Step tulajdonság által meghatározott mértékben módosítja a folyamatjelző Value tulajdonságát.

Megjegyzés: a folyamatjelző mérete tervezőnézetben a Properties munkaablak segítségével módosítható. Futásidőben használjuk a következő utasítást:

```
Folyamatjelzőnév.Size = New Drawing.Size(szélesség, magasság)
```

Vezérlőelemek futásidejű létrehozása, törlése

Egy vezérlőelem (beleértve az űrlapot is) Controls kollekciója 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 vezérlőelemobjektum As vezérlőelemtípus
   vezérlőelemobjektum = New vezérlőelemtípus
   ```
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()
   ```

Események

Esemény	Eseménynév	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 billentyűkódja (Keys típusú felsorolás)
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 megosztott metódust 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 felsorolástípusok:
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 hívása: New Timer(). Az eszköztárból is elhelyezhető az űrlapon.
Hivatkozás: *változónév*.*tulajdonságnév*, *változónév*.*metódusnév*()

Interval	A Tick események gyakorisága ezredmásodpercben.
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 megosztott tulajdonsága, például: System.Text.Encoding.{ASCII | Default | Unicode | UTF8 | …}
3. A fájl olvasása metódusokkal
4. A fájl lezárása:
   `változónév.Close()`
5. Az objektum felszabadítása:
   `változónév.Dispose()`

A StreamReader típusú objektum tulajdonságai és metódusai

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

EndOfStream	True, ha elértük az adatfolyam (stream) végét.
Close()	Lezárja az adatfolyamot. Meghívja a Dispose metódust.
Dispose()	Felszabadítja az adatfolyamhoz 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 az adatfolyam 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 az adatfolyamból. Visszatérési értéke Nothing, ha elértük az adatfolyam végét.
ReadToEnd()	Beolvassa az adatfolyam hátralévő részét. Ha már az adatfolyam végén állunk, akkor üres üres sztringet ("") ad vissza.

Alapértelmezett karakterkódolási mód: UTF-8.
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:

```
Do While fájlobjektum.Peek() <> -1 … Loop
```

A szövegfájlok egy sorában tárolt több értéket a sztring típus 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 megosztott tulajdonsága, például: System.Text.Encoding.{ASCII | Default | Unicode | UTF8 | …}
   Alapértelmezett karakterkódolás: UTF-8.
3. A fájl írása metódusokkal
4. A fájl lezárása (**elmulasztása adatvesztéssel járhat!**):
   `változónév.Close()`
5. Az objektum felszabadítása:
   `változónév.Dispose()`

A StreamWriter típusú objektum metódusai

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

Close()	Lezárja az adatfolyamot (streamet). Meghívja a Flush, majd a Dispose metódust.
Dispose()	Felszabadítja az adatfolyamhoz kapcsolódó erőforrásokat.
Flush()	Kiírja a puffer tartalmát az adatfolyamba, majd törli a puffert.
Write(*kifejezés*)	A *kifejezés* értékét (sztringként) az adatfolyamba írja.
WriteLine([*kifejezés*])	A *kifejezés* értékét (sztringként) a soremelés kódjával kiegészítve az adatfolyamba írja.

Alapértelmezett karakterkódolási mód: UTF-8
Megjegyzés: az adatvesztés elkerülése miatt az írás befejezése után mindenképpen célszerű meghívni a Flush, majd a Close metódust. A Close metódus hívása után nem hivatkozhatunk az adatfolyamra.

Az OpenFileDialog objektum tulajdonságai és metódusai

Fájl kiválasztásához használjuk.
Névtér: System.Windows.Forms
A konstruktor hívása: `New OpenFileDialog()`
Részletesebben lásd a *Programozási ismeretek* 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 az állományt!

A SaveFileDialog objektum tulajdonságai és metódusai

Fájl helyének és nevének megadásához használjuk. Részletesebben lásd a tankönyv Megnyitás és mentés párbeszédablakokkal című leckéjében.
Névtér: System.Windows.Forms
A konstruktor hívása: `New SaveFileDialog()`

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 FolderBrowserDialog objektum tulajdonságai és metódusai

Mappa kiválasztásához használjuk.
Névtér: System.Windows.Forms
A konstruktor hívása: `New FolderBrowserDialog()`

Description	A párbeszédablakban a mappalista felett megjelenő magyarázó szöveg.
RootFolder	A mappalista kiinduló eleme.
SelectedPath	A felhasználó által kiválasztott mappa elérési útja.
ShowNewFolder	True értéke esetén megjelenik az Új mappa parancsgomb (alapértelmezett érték: True).
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!

Egyszerűsített fájlkezelés (Using-blokk)

A fájlok megnyitása és lezárása a Using utasítással egyszerűsíthető.
A Using-blokk szerkezete:

```
Using erőforrások
  [utasítások]
End Using
```

Az *erőforrások* szintaxisa:

```
erőforrásnév As erőforrástípus = kifejezés[, …]
```

Például:

```
Using Fájlki As IO.StreamWriter = New IO.StreamWriter(elérési út)
  írás a fájlba
End Using
```

A Using-blokk garantáltan lezárja a fájlt, és felszabadítja az erőforrásokat, akárhogyan is lépünk ki belőle.

A fájlrendszer objektumai és megosztott metódusai

Az osztályok és objektumok használatánál ügyeljük a hozzáférési jogokra!
További fájlkezelési lehetőségeket szolgáltatnak a Microsoft.VisualBasic.FileIO.FileSystem objektumosztály és a My.Computer.FileSystem objektum metódusai.
Lásd még a My.Computer névtér ismertetését a 75. oldalon!

A DriveInfo objektum tulajdonságai

Névtér: System.IO
A konstruktor hívása: `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 megosztott metódusai

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(*elérésiút*)	True, ha létezik a megadott mappa, egyébként False.
Directory.GetCurrentDirectory()	Megadja az aktuális mappa elérési útját tartalmazó sztringet.
Directory.GetDirectories(*elérésiút*[, *minta*[, *keresés*]])	Sztringtömbként megadja a meghatározott *elérésiút* mappában található mappákat. A *minta* sztring tartalmazhat helyettesítő karaktereket (?, *). A *keresés* System.IO.SearchOption típusú felsorolás (TopDirectoryOnly: almappákban nem keres, AllDirectories: almappákban is keres).
Directory.GetFiles(*elérésiút*[, *minta*[, *keresés*]])	Sztringtömbként megadja a meghatározott *elérésiút* mappában található fájlokat. A *minta* és *keresés* argumentumok jelentését lásd a GetDirectories metódusnál!
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!).

Megjegyzés: a megosztott metódusok használata akkor hatékony, ha ugyanarra a mappára csak néhányszor hivatkozunk. Sokszori hivatkozás esetén alkalmazzuk inkább a DirectoryInfo osztály példánymetódusait.
Az aktuális mappára a My.Computer.FileSystem objektum CurrentDirectory írható/olvasható tulajdonságával is hivatkozhatunk.
Az aktuális felhasználó Dokumentumok mappájának elérési útja: My.Computer.FileSystem.SpecialDirectories.MyDocuments

A File objektumosztály megosztott metódusai

Névtér: System.IO
Elérésiút, forrás, cél: sztringkifejezés
Kódolás: a System.Text.Encoding osztály osztálytulajdonsága, például: System.Text.Encoding.{ASCII | Default | Unicode | UTF8 | …}

File.AppendAllLines(*elérésiút*, *sztringkollekció*[, *kódolás*])	Megnyitja a fájlt (létrehozza, ha nem létezik), hozzáfűzi a felsorolható *sztringkollekció* tartalmát (soronként), majd lezárja a fájlt.
File.AppendAllText(*elérésiút*, *szöveg*[, *kódolás*])	Megnyitja a fájlt (létrehozza, ha nem létezik), hozzáfűzi a szöveg-et, majd lezárja a fájlt.
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.GetCreationTime(*elérésiút*) File.GetLasAccessTime(*elérésiút*) File.GetLastWriteTime(*elérésiút*)	A fájl létrehozásának, utolsó megnyitásának, illetve utolsó módosításának dátuma/időpontja.
File.Move(*forrás*, *cél*)	Átmozgatja a forrásfájlt.
File.ReadAllLines(*elérésiút*[, *kódolás*])	Megnyitja a szövegfájlt, beolvassa a teljes tartalmát, majd lezárja a fájlt. Visszatérési értéke sztringtömb, melynek elemei a fájl sorait tartalmazzák.
File.ReadAllText(*elérésiút*[, *kódolás*])	Megnyitja a szövegfájlt, beolvassa a teljes tartalmát, majd lezárja a fájlt. Visszatérési értéke egyetlen sztring.
File.ReadLines(*elérésiút*[, *kódolás*])	Megnyitja a szövegfájlt, beolvassa a teljes tartalmát, majd lezárja a fájlt. Visszatérési értéke sztringkollekció, melynek elemei a fájl sorait tartalmazzák. A kollekció feldolgozását elkezdhetjük akkor is, amikor még nem fejeződött be teljesen a fájl beolvasása (például nagyon nagy méretű fájlok esetén).
File.SetCreationTime(*elérésiút*) File.SetLasAccessTime(*elérésiút*) File.SetLastWriteTime(*elérésiút*)	Módosítja a fájl létrehozásának, utolsó megnyitásának, illetve utolsó módosításának dátumát/időpontját.
File.WriteAllLines(*elérésiút*, *sztringkollekció*[, *kódolás*])	Létrehoz egy új fájlt (felülírja a létező fájlt), beleírja a felsorolható *sztringkollekció* tartalmát (soronként), majd lezárja a fájlt.
File.WriteAllText(*elérésiút*, *szöveg*[, *kódolás*])	Létrehoz egy új fájlt (felülírja a létező fájlt), beleírja a *szöveg*-et, majd lezárja a fájlt.

Megjegyzés: a megosztott metódusok használata akkor hatékony, ha ugyanarra a fájlra csak néhányszor hivatkozunk. Sokszori hivatkozás esetén alkalmazzuk inkább a FileInfo osztály példánymetódusait!

A Path objektumosztály megosztott metódusai

Névtér: System.IO
Elérésiút: sztringkifejezés

Path.ChangeExtension(*elérésiút*, *kiterjesztés*)	Visszaadja az *elérésiút* sztringet, de lecseréli benne a kiterjesztést a (ponttal vagy pont nélkül) megadott *kiterjesztés*-re. *Kiterjesztés* = Nothing esetén törli az *elérésiút*-ból a kiterjesztést (ha van).
Path.GetDirectoryName(*elérésiút*)	Visszaadja az *elérésiút* sztringet, de a fájlnév nélkül.
Path.GetExtension(*elérésiút*)	Visszaadja az *elérésiút*-ban szereplő kiterjesztést.
Path.GetFileName(*elérésiút*)	Visszaadja az *elérésiút*-ban szereplő utolsó elemet (mappa- vagy fájlnevet) az esetleges kiterjesztéssel együtt.
Paht.GetFileNameWithoutExtension(*elérésiút*)	Visszaadja az *elérésiút*-ban szereplő fájlnevet az esetleges kiterjesztés nélkül.
Path.GetFullPath(*elérésiút*)	Megadja az *elérésiút*-hoz tartozó abszolút elérési utat.
Path.HasExtension(*elérésiút*)	True, ha az *elérésiút* tartalmaz kiterjesztést.

Grafika

A grafikai objektumok számos további tulajdonsággal és metódusokkal rendelkeznek. Részletesebben lásd a Visual Basic súgóját.

A Pen (toll) objektumosztály

Névtér: System.Drawing
Objektumait vonalak rajzolásához használjuk.
A konstuktor hívása: 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.

A Brush (ecset) objektumosztályok

Az ecsetobjektumokat 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.

Grafikus metódusok

A grafikus metódusok sokféle változattal rendelkeznek. A következő táblázat egy-egy példát mutat a paraméterezésre.

Clear([*szín*])	Törli a rajzfelületet, és kitölti a megadott színnel.
DrawArc(*toll*, *x*, *y*, *szélesség*, *magasság*, *kezdőszög*, *végszög*)	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 (megosztott metódus).

A Bitmap (bitkép, bittérkép) objektumosztály

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

`New Bitmap(`*`képobjektum`*`)`: az Image/Bitmap típusú képobjektumból létrehozza az új 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(*bitkép*) megosztott metódusával hozunk létre és rendelünk hozzá a Bitmap objektumhoz.
A Bitmap objektumot az Image tulajdonság segítségével rendelhetjük hozzá egy vezérlőelemhez: *vezérlőelem*.Image = *bitképobjektum*
Automatikus típuskonverzió hiányában használjuk a CType konverziós függvényt: *vezérlőelem*.Image = CType(*bitképobjektum*, 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). Többek között 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* osztályok segítségével adhatjuk meg. 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 készítsünk róla másolatot. Ezután az eredeti Bitmap objektumot már felszabadíthatjuk:

```
Dim Temp, Rajz As Bitmap
Temp = New Bitmap(elérésiút)
Rajz = New Bitmap(Temp) ' átmásolja 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. Megosztott metódus!
Image.FromHbitmap(*azonosító*)	Elkészíti a megadott bitkép másolatát. Megosztott metódus! Az eredeti bitkép a metódushívás után már felszabadítható. Az azonosító például a Bitmap.GetHbitmap() metódussal határozható meg.
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* osztályok segítségével adhatjuk meg. Részletesebben lásd a súgóban!

Megjegyzés: a projekt erőforrásfájljai (Project resource files) közé felvett kép hozzárendelése egy képet megjelenítő objektumhoz (például PictureBoxhoz):

```
objektumnév.Image = My.Resources.erőforrásnév
```
(az intelligens súgó kilistázza az elérhető erőforrásneveket), vagy:
```
objektumnév.Image = My.Resources.ResourceManager.GetObject(sztringkifejezés)
```
ahol a *sztringkifejezés* az erőforrás neve sztringként megadva

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 megjelennek.
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 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 futtatórendszer 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 futtatórendszer 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 röviden.

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 résztvevő 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 kiválasztja a lekérdezés eredményébe kerülő mezőket:

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

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 azonban tetszőleges. Az Aggregate kivételével bármely záradék elhagyható.
A lekérdezés eredményét egy felsoroló (IEnumerable) objektumban tároljuk. A névtelen (anonymous) típus alkalmazásakor a futtatórendszer 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 (összesítő) záradék

Összesítő 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 futtatórendszer 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: összesítő 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 összesítő és csoportosító 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
MS Office 2010 esetén:
– Microsoft Word 14.0 Object Library
– Microsoft Excel 14.0 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

[7] Component Object Model

Egyéb elemek

Véletlenszám-generálás

A Random objektumosztály objektumai egyenletes eloszlású véletlenszámokat generálnak.
Névtér: System
A konstruktor hívása:

– 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!

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 megosztott tulajdonságai és metódusai

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 adatfolyamból olvas, a Write/WriteLine a standard output adatfolyamba ír.

A DateTimePicker objektumosztály

Névtér: System.Windows.Forms
A konstruktor hívása: `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 a vezérlőelemeknél feltüntetett általános tulajdonságokkal és metódusokkal is rendelkezik (lásd ott).
Megyjegyzés: dátumokat a MonthCalendar vezérlőelemmel szintén bekérhetünk. A MonthCalendarral egyszerre két hónapot jeleníthetünk meg, így hosszabb időtartam választható ki.

A Stopwatch objektumosztály

Névtér: System.Diagnostics
A konstruktor hívása: `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	True, ha a stopper éppen időt mér (a Start és a Stop között vagyunk).
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 egy 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, ami miatt egyes szerverek hibaüzenettel válaszolnak a kérésre.
Network.UploadFile(*forrásfájl*, *URL*[, …])	Feltölti a megadott forrásfájlt. A további argumentumok megfelelnek a DownloadFile metódus argumentumainak.

Előre definiált konstansok

vbNewLine	új sor
vbTab	tabulátor

TriState felsorolás:	
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.

Megosztott 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 (billentyűkódok)

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