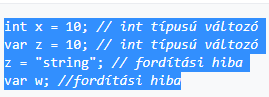
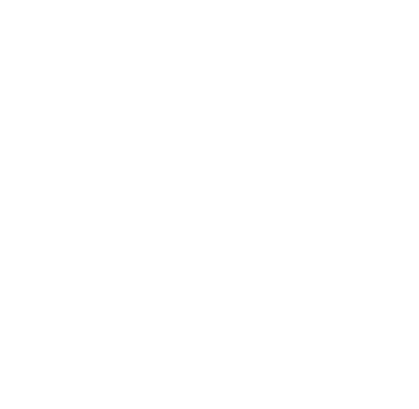
***ADATTÍPUSOK***

**Egész Típusok:**

 Sbyte(*előjeles*) 1 byte

Byte 1 byte(*csak + számok*)

Ushort 2 byte(*csak + számok*)

Short 4 byte

Uint 4 byte(*csak + számok*)

Int 2 byte

Long 8 byte

ULong 8 byte(*csak + számok*)

**Valós típusok:**

Float 4 byte

Double 8 byte

Decimal 16 byte

**Szöveges típus:**

Char 1 karakter

String 0-2 milliárd karakterig

**Logikai típus:**

Boolean 2 byte

**Egyéb :**

Bool true vagy false

1. A C# erősen típusos nyelv, ami következőt jelenti: minden változó típusának ismertnek kell lennie a fordítási időben. A változó típusa határozzam meg, hogy az milyen, és mekkora értéket vehet fel, ill. hogy mennyit foglal a memóriából.  
     
   ***Read more:***[***http://prog-tech.hupont.hu/3/adattipusok-valtozok#ixzz7k2wsIDld***](http://prog-tech.hupont.hu/3/adattipusok-valtozok#ixzz7k2wsIDld)

bool Contains(string s)

Igaz értéket ad vissza, ha a paraméterként megadott szövegrészletet tartalmazza a szöveg.

bool EndsWith(string s)

Igaz értéket ad vissza, ha a szöveg a megadott szövegrészlettel végződik.

int IndexOf(char c)

int IndexOf(String s)

A paraméterként megadott karakter vagy szövegrészlet első előfordulásának indexét adja vissza.

int LastIndexOf(char c)

int LastIndexOf(String s)

A paraméterként megadott karakter vagy szövegrészlet utolsó előfordulásának indexét adja vissza.

int Length;

A szöveg hosszának lekérdezésére szolgáló tulajdonság.

string PadRight(int totalWidth)

string PadRight(int totalWidth, char paddingChar)

Egy új szöveget hoz létre, amely a paraméterként megadott számú karakterből áll. A megadott szöveg szélességen belül úgy igazítja a szöveget, hogy az balra igazított legyen, tehát az eredeti szöveg utolsó karaktere után annyi szóközt illeszt (jobb oldalról bővíti), hogy a szöveg a paraméternek megadott számú karakterből álljon. Kétparaméteres változatában a második paraméter a kitöltő karaktert határozza meg.

string PadLeft(int totalWidth)

string PadLeft(int totalWidth, char paddingChar)

Hasonló a PadRight metódushoz. Ez azonban a szöveget jobbra igazítja, tehát a szóközökkel való feltöltés az eredeti szöveg első karaktere előtt kezdődik meg (bal oldalról bővíti). A kétparaméteres változatban a második paraméter a kitöltő karaktert határozza meg.

string Remove(int startIndex)

string Remove(int startIndex, int count)

A megadott index pozíciótól kezdve törli a szöveg karaktereit és az eredményt egy új szövegként adja vissza. Kétparaméteres változatában a második paraméter a törlendő karakterek számát határozza meg. Az egyparaméteres változat a megadott indextől a szöveg végéig töröl.

string Replace(string oldValue, string newValue)

string Replace(char oldChar, char newChar)

Az első paraméter által megadott szövegrészlet vagy karakter cseréje a szövegben a második paraméter által meghatározott szövegrészletre vagy karakterre. Az eredményt egy új szövegként adja vissza.

string[] Split(params char[] separator)

A megadott elválasztó karakterek alapján a szöveget felszeleteli szövegekből álló tömbre.

bool StartsWith(string value)

Igaz értéket ad vissza, ha a szöveg a megadott szövegrészlettel kezdődik.

***Egyszerű adattípusok***

Az egyszerű adattípusok esetén a tárolt adat tovább nem bontható részekre.

Egyszerű adattípusnak számítanak a logikai, karakter, egész és valós típusok.

A sztring típus nehezen eldönthető kategória. Mivel tovább bontható részekre – karakterekre. Azonban ha nem akarjuk felbontani, akkor egy szöveg egy egységet alkot. Ennek megfelelően a sztring típus egyes esetekben egyszerű adattípusnak is mondható.

***Összetett adattípusok***

Összetett adattípus esetén több adat tárolását feltételezzük.

Ezen több adat lehet több elemi típusú adat, de lehet több összetett típusú adat is, sőt, akár vegyes is lehet.

Összetett adattípusnak minősül a sztring típus, amennyiben karakterenként kezeljük (a sztring több elemi adattípus, a karakterből felépített összetett adattípus).

Hasonlóan összetett adattípusnak minősül a rekord, a tömb, a sor, a verem, a lista, és még számtalan más is.

***Összetett adattípusok tárolás szerint***

Összetett adattípus esetén több adat tárolását feltételezzük.

Amennyiben ezen adatok mindegyike egyforma típusú, úgy ’homogén’-nek nevezzük az összetett adatszerkezetet. Ilyen például a tömb.

Amennyiben ezen adatok különböző típusúak, úgy ’inhomogén’, vagy ’heterogén’ adatszerkezetről beszélünk. Erre a legjobb példa a rekord.

Amennyiben több elem a memóriában egymást követő területen helyezkedik el, úgy folytonos reprezentációjúnak nevezzük. Ilyen a rekord is, a tömb is.

Amennyiben az elemek a memóriában független területeken (elszórva) helyezkednek el, úgy szétszórt reprezentációjúnak nevezzük. Ilyen a klasszikus sor adatszerkezet.

Amennyiben több adat tárolása esetén a darabszámot előre be kell jelenteni a deklaráció során, úgy statikus helyfoglalású összetett adatszerkezetről beszélünk. Ilyen a rekord és a tömb is.

Amennyiben előre nem kell bejelenteni a darabszámot, hanem az futás közben szabadon változtatható, úgy dinamikus méretű adatszerkezetről beszélünk.

***Összetett adattípusok kezelés szerint***

Összetett adattípus esetén több adat tárolását feltételezzük.

A több adat esetén beszélni kell az adathozzáférési sebességről, stratégiákról.

Amennyiben ezen adatok bármelyikének elérése egyforma sebességgel történik, úgy ’véletlen elérésű’ (Random Access) adatszerkezetről beszélünk (ilyenek a tömbök és a rekordok például).

A klasszikus lista esetén a listában hátrébb lévő elemek elérése tovább tart, mint az előrébb lévő elemeké. Az ilyen adatszerkezetet szekvenciális elérésűnek nevezzük.

A SOR adatszerkezet speciális adatelérési technikáját FIFO-nak (First In First Out) nevezzük.

A VEREM adatszerkezet speciális adatelérési technikáját LIFO-nak (Last In First Out) nevezzük.