







# VEREM (STACK)

*A verem megengedett o xgngvgk*

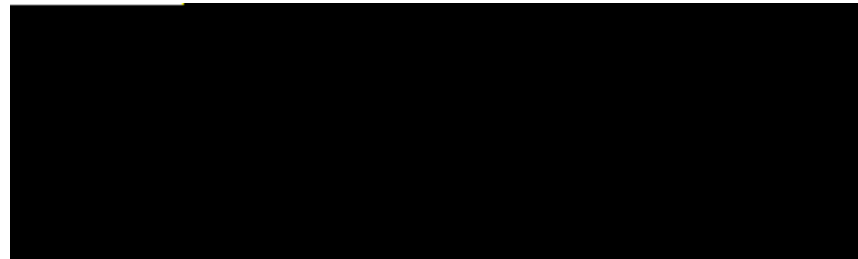
- **Verembe** (*push*): egy elem betétele a verembe (a verem „tetejére”).  
Csak akkor lehetséges, ha a verem még nem telt meg.
- **Veremb**

# VEREM (STACK)

*Xgtg o " o g i x c n » u v <sup>a</sup> u c*

## Folytonos tárolás

A vermet vektorral kezeljük, ahol az elem mindig a vektor .  
Bovítani, új elemet felvinni csak a verem tetejéhez lehet. Hozzáférti is csak a verem tetejéhez lehet. Ha egy közben elemet karunk  
kkor törölni ll lette lévo lemket







```
//Verem létrehozása  
Stack<int> verem = new Stack<int>(); <
```

```
//Elemek hozzáadása
```

```
verem.Push(4); <  
verem.Push(6); <  
verem.Push(6); <
```

```
//Összeg kiszámítása
```

```
int osszeg = 0; <  
int osszeadando = 0; <
```

```
while(verem.Count > 0) <  
{ <
```

```
{osszeadando} "
```





# SOR (QUEUE)

*Uqto xgngvgm"*

- **Sorba** (*put*): egy elem betétele a sorba (a sor „végére”). Csak akkor lehetséges, ha a sor még nem telt meg.
- **Sorból** (*get*): a sor „elején” található elem kivétele. Csak akkor tudunk elemet kivenni, (kivenni)] IEBC 0904-VEF311 01 18210 (Gha)] IEQ
- **Inicializálás**: (*init*): alapállapotba helyezés, ürítés.
- **Üres-e**:

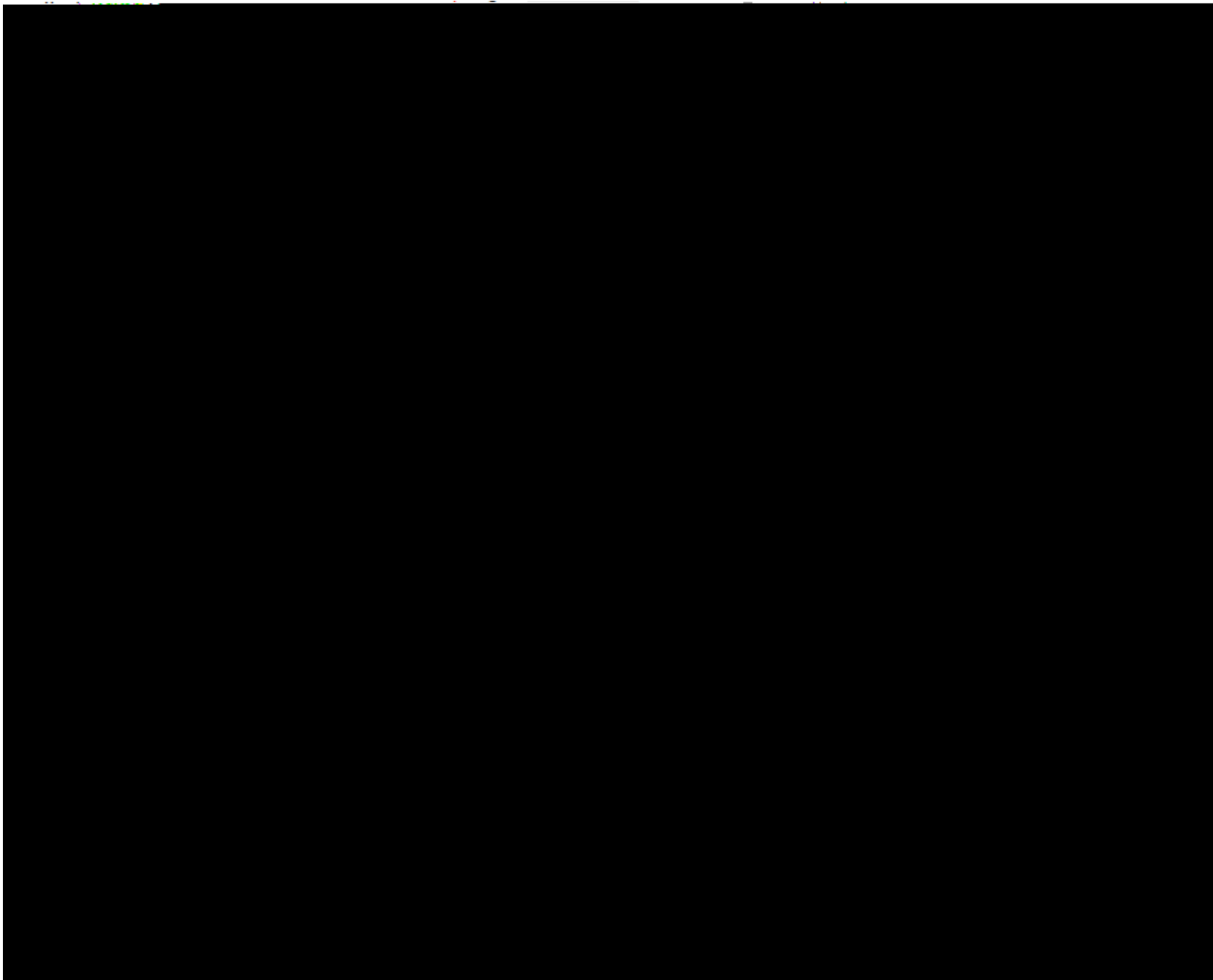


# **SOR (QUEUE)**

*Statikus sor*

Statikus megvalósítás esetén a sorba tett

**SOR**









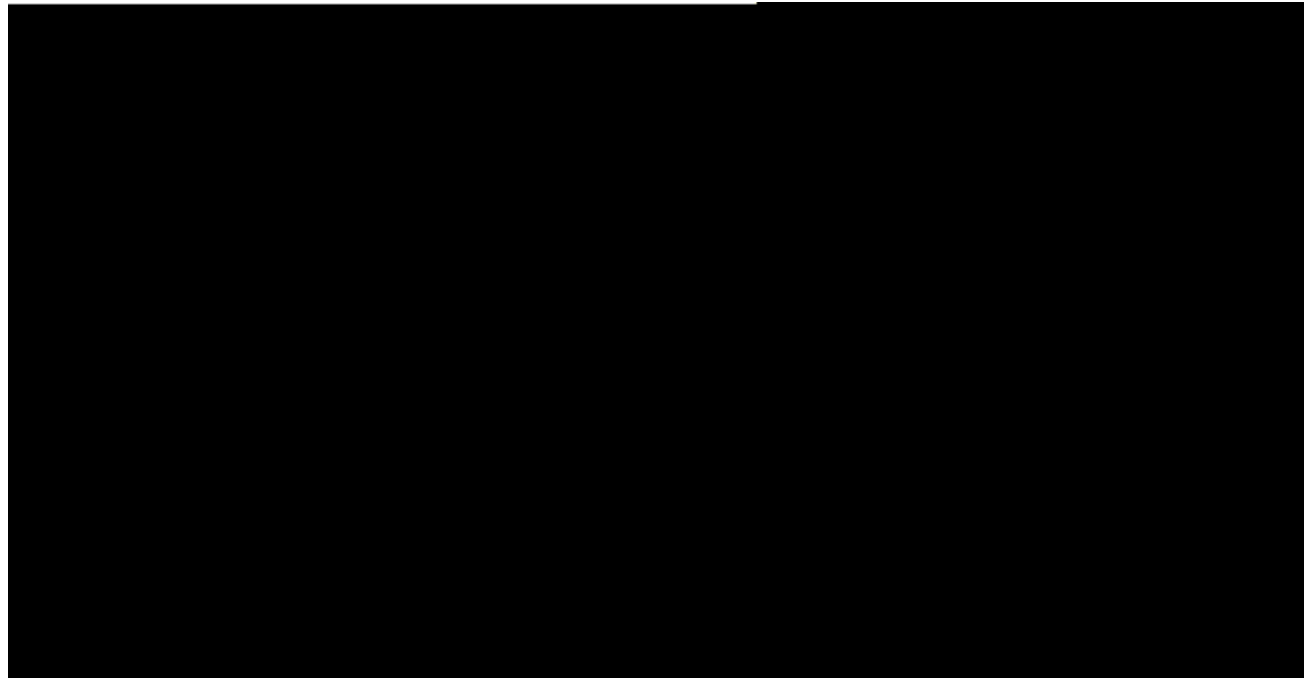






A lista első elemének eléréséhez csupán egy mutató is elegendő. Ennek ismerete a teljes lista ismeretét jelenti, mert így hozzáférhetünk az elemekben tárolt minden

Új elem felvitele  
Új elem felvitelét





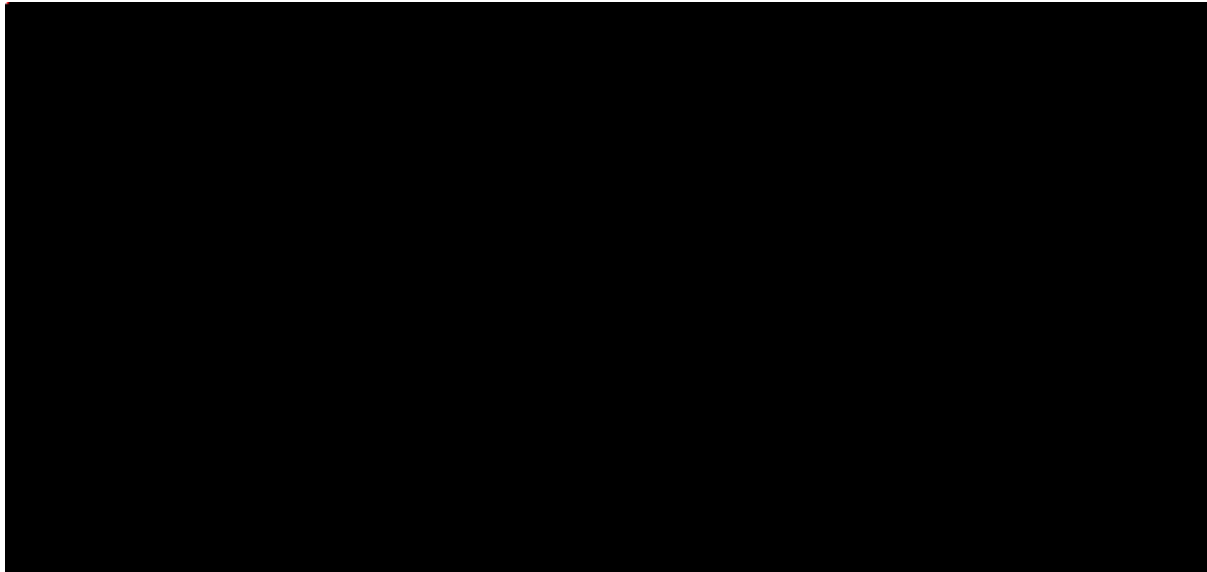








# Két irányban láncolt lista



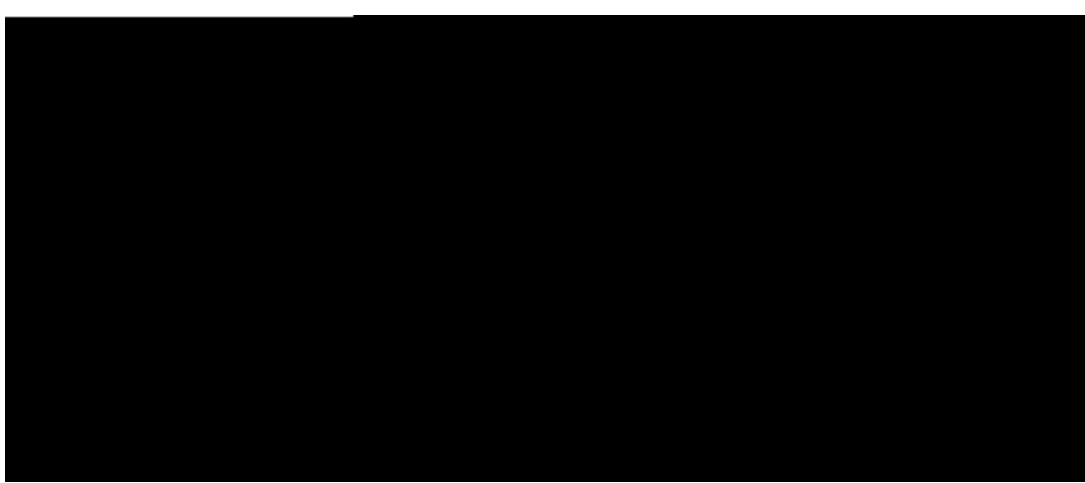
A -adatszerkezetek szervezéséből adódóan a



# LIST, M (generikus) c#-ban

készítette: Vastag Atila

2017



```
template <class T>
ostream& operator<<(ostream &
out, const List<T> &l){
    out<<endl;
    out<<"-----"<<endl;
    for(int i=1;i<=l.size();i++){
        if(i!=1) out<<" ";
        T res;
        l.rea
```



# LISTA példányosítása

List



Egy elem hozzáadása a lista végére

List<

Egy elem beszúrása a listába

List<int>

Liasta hozzáadása a lista végére

(J\ HOHP EHV]~UiVD D OLVWiED

List < int >

# Lista kiírása

```
using System;  
using System.Collections.Generic;  
class Program  
{  
    static public void Main()  
    {  
        List<int> intList = new List<int>();
```

# Lista kiírása

## List függvényei

```
void Clear()
```





```
static void Main(string[] args)
{
    List<dynamic> dynamicList = new List<dynamic>()
    {
        1,
        6.66,
        'a',
        "bla",
        true
    };

    Console.WriteLine("A dinamikusan lista elemei: ");

    foreach(dynamic listaElem in dynamicList)
    {
        Console.WriteLine($"{listaElem}\t\t{listaElem.GetType()}");
    }

    Console.ReadKey();
}
```

A *Dictionary* kulcs-érték párokat

A szótár elempárok tárolására szolgál, melyek közül egyik a kulcs (**TKey**), amely azonosítja az elempárt, másik az érték.

Minden kulcs egyedi.

Gyakorlatilag a szótár úgy viselkedik, mint egy lista, de az elemek indexe1] TETQ EMC /P <</MCID (3**TKey**)C q0.0000010729

Konstruktor gene109.riparaméter nélküli:

Dictionary(): létrehoz egy szótárt,



Random



1 - Egy számokat tartalmazó listát töltünk fel random számokkal, úgy hogy a program

5 – Két számokat tartalmazó listát töltünk fel random s  
program kezel je a program elejé megadja, hogy h elemre lehet számítani  
(





12 – A felhasználónak meg kell adnia két



Program "\*" végjelig kérje be egy kézilabda a csapat tagjainak nevét, de max 20-

Egy forgalom ellen rz ponton "n" számú autó haladt z a program \* végjelig





Program végjelig kérje be a kosárlabda csapat tagjainak nevét



