**A képen szöveg, képernyőkép, szám, sor látható

Automatikusan generált leírásTételek**

A képen szöveg, nyugta, képernyőkép, szám látható

Automatikusan generált leírás

A képen szöveg, nyugta, képernyőkép, szám látható

Automatikusan generált leírás**Szintaktika**

**Rekordok:**

public struct <név>{

Public <típus> <név>;

…

}

**Definiálás:**

<struct típus> <név> = new <struct típus>();

**Fgvkhez:**

rublic static <visszatérési típus> < név>(<arg1 tip> <arg1 név>, …){

…

return <visszateresi tipusnak megfelelo ertek>

}

**Átalakítás, konverzió:**

int.Parse(<string vált.>);

**Tömbök:**

<típus>[] <név> = new <típus> [<elemszámot meghatározó értéket tároló változó neve, vagy konstans>];

**Értékkel deklarálás:**

<típus>[] <név> = {elem1, elem2,...};

**Beolvasás több adatot soronként:**

string sor = Console.ReadLine();

tömb[i]:=int.Parse(sor.Split(’ ’)[j]);

**További infok:**

//mainen belul 0-ra inicializalja a C# a tömböket

**Listák:**

List<<típus>> <név> = new List<<típus>>();

**Műveletek:**

<listanév>.Add(<érték>);

-||-.Remove(-||-);

<lista>.Count; //hosszlekérdezés

//Indexelés is működik

**Mátrixok:**

<típus>[,] <név> = new <típus>[<sorok>,<oszlopok>]

**Stringösszefűzés (kííratáshoz)**

+ operátorral, intet implicit konvertál stringgé

**Rendezés algo:**

public void minkivrendez(int[] tomb){

int min;

for (int i=0; i<tomb.GetLength(0)-1; i++){

min=i;

for( int j=i+1;j<tomb.GetLength(0); j++){

if (tomb[min]>tomb[j]){

min=j;

}

Csere(tomb[i], tomb[min]);

}

}

**Tömb átadás paraméterként:** evidens módon

**Tömb méret lekérdezés:**

<arr>.GetLength(<dimenzio szama>);