tahák - 2.cviko

Pole (array)

C# má pole pevné délky, kterou si rozmyslíte dopředu a nebo rovnou při konstrukci pole explicitně uvedete:

```
int[] poleCisel = new int[23]; //pole intu delky 23, vsude jsou nuly
int[] poleVyplnene = new int[]{4,5,6,8};
```

Pole můžete vytvářet libovolného typu.

Vícerozměrná pole

Pokud potřebujete vícerozměrné pole, máte dvě možnosti:

2-D array

dvoudimenzionální pole (matice), vlastně je to obdelnik mxn.

```
int[,] matice = new int[m,n];
```

Jagged array (zubaté pole, pole polí)

Toto pole je v podstatě jen pole, jehož prvky jsou také pole. Každý prvek tak může být jinak dlouhý. Při vytváření stačí udat počet polí/řádků:

```
int [][] jagg = new int[3][];
```

vnitřní pole/řádky pak musíme vytvořit samostatně:

```
jagg[0] = new int[3];
jagg[1] = new int[5];
jagg[2] = new int[2];
jagg[3] = new int[8];
jagg[4] = new int[10];
```

Dynamicky alokovaná pole (List)

Chceme-li použít pole proměnné délky (= list), musíme použít knihovnu System.Collections.Generic; kde tato pole bydlí:

```
using System.Collections.Generic;
List<int> listCisel = new List<int>();
List.Add(3); //pridame jeden prvek
```

Do `<>` závorek píšeme typ, který bude v poli uložen. List je tzv. generická třída, podrobněji to budeme řešit později, v zásadě to znamená, že existuje třída List pro každý typ, který tam lze uložit (List čísel, List stringů, List objektů, List floatů...). Na to, aby vývojáři nemuseli psát každou třídu znova, slouží právě generické třídy.

Kdy použít list a kdy pole?

U pole je třeba si dopředu rozmyslet, jak bude cca velké (sem tam nějaká položka navíc nás netrápí, zvlášť když je to třeba pole čísel). Při inicializaci pole se v paměti rovnou vytvoří místo pro celé pole.

List je vlastně pole pevné délky, které se ale v případě potřeby zdvojnásobí. Z toho plyne několik problémů:

- nemáme dopředu místo v paměti
- nevíme jestli tam bude místo
- chceme-li (a to velmi často chceme) mít všechny prvky pole u sebe, je třeba uložená data přeskládávat, aby se nám tam nové (zvětšené) pole vešlo
- musíme překopírovat prvky do nového většího pole

Z toho plyne, že používání polí je rychlejší než používání listů a pokud nám nedá příliš práce zjistit rozumný horní odhad na pole, je lepší použít pole.