03 - Strukturované datové typy

- objekty, které se skládají z několika členů
- k jednotlivým členům lze přistupovat pomocí selektorů
- **Homogenní** - komponenty mají stejný typ
- **Heterogenní** - komponenty mají rozdílný typ

Objekt

- abstraktní datový typ
- obsahuje vlastnosti, funkce
- základní stavební kámen OOP
- obsahuje možnost zapouzdření
- existující objekty mohou být v programu uprovovány

```
public class User
{
  public int Id {get; set;}
  public string FirstName {get; set;}
  public string LastName {get; set;}
  private string Password;

  public override string ToString()
  {
    return $"({Id}) User's full name: {FirstName} {Lastname}";
  }
}
```

Struktura (struct)

- skupina více proměnných dohromady (mohou mít různé datové typy)
- hodnotový typ
- nepodporuje dedičnost, nemá konstruktor

```
struct Souradnice
{
  int x;
  int y;
  string pozice;
}
```

Pole

- homogenní datový typ
- velké množství proměnných stejného datového typu
- oproti listu má **pevný počet položek**, který nelze měnit
- každá položka je označena indexem (začínají od 0)
- skupina více proměnných dohromady (mohou mít různé datové typy)

```
int[] pole = new int[50];
pole[0] = 4;
Console.WriteLine($"První položka z pole: {pole[0]}");
```

Vícerozměrná pole

- jde o pole, ve kterých se nachází další pole
- můžou být N-rozměrná
  - nejčastěji jsou ovšem 2D

Kolekce

- soubor dat, většinou stejného typu - ke specifickému účelu
- slouží jako úložiště objektů a zajištění přístupu k nim
- list, pole, seznam, strom, slovník, zásobník, fronta…

Generické kolekce

- lze specifikovat jejich datový typ až ve chvíli, kdy se vytváří instance
- obecné kolekce
- silně typované
- např. list v C#:

```
List<string> listp = new List<string>();
listp.Add("item");
Console.WriteLine($"První položka z pole: {listp[0]}");
```

Negenerické kolekce

- přebírají jakýkoliv datový typ
- nejsou silně typované
- např. fronta v C#:

```
Queue fronta = new Queue();
fronta.Enqueue(1);
fronta.Enqueue(2);
fronta.Dequeue();
```

Čtení prvků z kolekce

- cyklus foreach iteruje přes kolekce
- lze vytvořit i vlastní enumerator pomocí implementování příslušného rozhraní

```
List<string> listp = new List<string>();
foreach(string item in listp)
{
  Console.WriteLine(item);
}
```