Programování

Konstanta, Enum, Struct, rekurze

Konstanta - const

- V matematice, fyzice, chemii a dalších oborech je konstanta pevně dané číslo
- V programování chápeme konstantu jako proměnnou, která se v průběhu programu nemění
  - Konstanta může být libovolného datového typu
- Konstanty knihovny Math:

```
public const double PI = 3.1415926535897931;
public const double E = 2.7182818284590451;
```

- Doporučení pojmenovávat konstanty velkými písmeny

```
const int CISLO = 123;
const string PREFIX = "_X";
const double DESETINNE = 11.8;
```

Výčtový typ - enum

- = typ hodnoty definovaný sadou pojmenovaných konstant reprezentované jako celočíselná hodnota int
- Vhodné pokud chceme vytvořit výčet hodnot, které se budou v programu objevovat
- Přestože si jednotlivé proměnné pojmenujeme podle naší potřeby, kompilátor je čte jako hodnoty int

```
public enum Den { Pondeli, Utery, Streda, Ctvrtek, Patek }
```

- Hodnoty jsou podle pořadí reprezentovány číslicemi od 0
- Pokud chceme můžeme reprezentované hodnoty změnit případně posunout

```
public enum Den { Utery = 2, Streda, Nedele = 7 }
```

- Volání výčtového typu: `Den.Utery`

Stukura - struct

- Hodnotový typ, který zapouzdřuje jiné hodnoty
- Vhodné pro reprezentaci jednoduchých objektů
- Nemají zbytečnou složitost oproti objektům vytvořené z třídy
- Výhodou je vyšší rychlost při manipulaci oproti třídám
- Pro získání a úpravu hodnot jednotlivých položek struktur využíváme příslušných funkcí – **get**, **set**
  - Položky struktury musí začínat velkým písmenem
- Oproti proměnné nemáme možnost nejprve strukturu deklarovat a následně ji přiřadit hodnotu
- Pokud chceme vytvořit proměnnou, která bude obsahovat strukturu využijeme tzv. **konstruktoru**, který nám strukturu vytvoří a vyplní příslušnými hodnotami
- Struktury mohou mít své vlastní vnitřní funkce

Stuktura - struct

```
public struct Bod
{
    Počet odkazů: 1
    public Bod(int x, int y)
    {
        this.X = x;
        this.Y = y;
    }
    Počet odkazů: 4
    public int X { get; set; }
    Počet odkazů: 2
    public int Y { get; set; }
    Počet odkazů: 0
    public bool PrusecikOsy()
    {
        return X == 0 || Y == 0;
    }
}
```

Konstruktor struktury

Výčet datových typů, které tvoří strukturu

Vlastní funkce struktury

Rekurze

- Obecně se jedná o stav, kdy objekt obsahuje sám sebe
- **Rekurzivní program** = program, který volá sám sebe
- Rekurzivní algoritmus vzniká tehdy, když rozdělením algoritmu na menší části nám vznikne stejný algoritmus, který řeší menší rozsah vstupních dat
- Nejčastěji se s rekurzí setkáváme u funkcí
- Přímá rekurze – funkce volá sama sebe
- Nepřímá rekurze – funkce volá jinou funkci, která opět volá funkci původní
- Lineární rekurze – funkce se v jednom průchodu zavolá jen jednou
- Stromová rekurze –funkce se v jednom průchodu zavolá vícekrát

Rekurze

- Pozor na zacyklení !!!
- U každé rekurze musíme dávat pozor, aby počet opakování nebyl příliš velký nebo dokonce nekonečný
- Počet volání = **hloubka rekurze**
- Při konstrukce funkce využívající rekurzi musí vždy dojít ke zmenšení rozsahu problému řešení
- Nutná je ukončující podmínka pro rekurzi
- Rekurze nemusí být vždy vhodné řešení kvůli paměťové náročnosti
- Pokud lze využít iterativního cyklu (for cyklus) použijeme jej místo rekurze
- V obecnosti platí, že každý iterativní algoritmus lze zapsat rekurzivně – platí i obráceně
- Fibonacciho posloupnost – lze řešit rekurzivně, ale pro paměťovou náročnost je lepší použít iterativního řešení