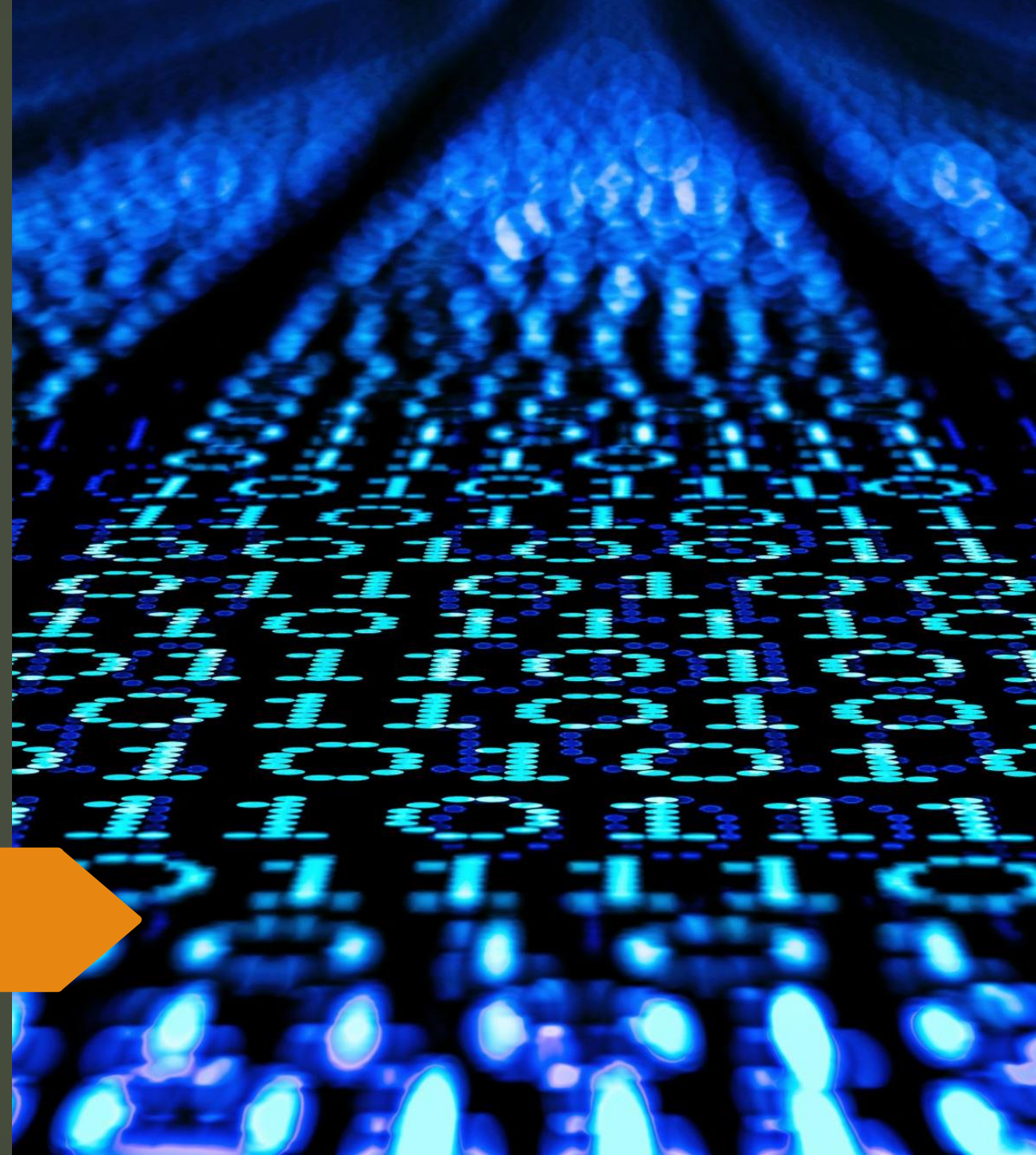


Programování

Podprogram (metoda)





Podprogram (metoda)

- ▶ = samostatná část programu, kterou můžeme opakovaně volat na různých místech
- ▶ V implementaci je podprogram reprezentován metodou/funkcí
- ▶ Slouží pro rozdělení celého algoritmu do menších částí pro lepší přehlednost a jednodušší správu
- ▶ Při vhodném použití redukuje nevyžádanou duplicitu
- ▶ U již definovaných metod nás nezajímá vnitřní logika, ale důležité je pro nás její použití

Struktura metody

```
public double Foo(int a, double b, out string c)
{
    if(a <=0) return -1;

    c = $"{b}^{a}";


    return Math.Pow(b, (double) a);
}
```

Hlavička metody

Tělo metody
ohraňené { }

```
string prikklad;
Foo(10, 1.8, out prikklad);
```

Volání metody



Návratová hodnota metody

- **Metody bez návratové hodnoty**
- Datový typ je nahrazen klíčovým slovem **void**
- Přestože funkce nic nevrací, můžeme vynutit její ukončení pomocí **return**
- **Metody s návratovou hodnotou**
 - Vždy musí obsahovat klíčové slovo **return**
 - Metody mohou vracet:
 - Základní datové typy
 - Struktury
 - Kolekce
 - Výčtový typ
 - Objekty



Parametry metod

- **Vstupní**
 - Parametry volané hodnotou – konkrétní nebo v proměnné
- **Výstupní**
 - Parametry volané odkazem na proměnnou s klíčovým slovem **out**
- **Vstupně-výstupné**
 - Parametry volané referencí na proměnnou klíčové slovo **ref**
- **Formální parametry**
 - Vnitřní pojmenování proměnné v implementaci funkce
- **Skutečné parametry**
 - Hodnoty použité při volání funkce
- **Null-able parametry**
 - Umožňuje předat metodě hodnotu null

Parametry metod – ukázka použití

```
public void Funkce(int promB) {  
    promB = 12;  
}  
int promB;  
Funkce(promB);
```

```
public void Funkce(ref int promC) {  
    promC = promC + 12;  
}  
int promC = 10;  
Funkce(promC);
```

```
public void Funkce(out int promA) {  
    promA = 12;  
}  
  
int promA;  
Funkce(out promA);
```

```
public void Funkce(int? promD){  
    if(promD == null)  
        return;  
    else Console.WriteLine(promD*10);  
}  
Funkce(null);  
Funkce(3);
```

Přetížení metod

- Stejně jako u proměnných nelze vytvářet dvě stejné funkce
- Lze ovšem vytvořit funkce se stejným názvem, ale jinými vstupními parametry
=> **přetížení metody**
- **Přetížení metody není ekvivalentní k přepisování metody**

```
public double Prumer(int a, int b, int c) {  
    return (a+b+c)/2.0;  
}
```

```
public double Prumer(double a, double b) {  
    return (a+b)/2;  
}
```

```
public double Prumer(int a, int b) {  
    return (a+b)/2.0;  
}
```

Rekurzivní metoda

- = funkce, která volá sama sebe
- Abychom se nezacyklili je třeba myslet na následující:
 - Následující volání musí pracovat s menším objemem dat nebo hodnotou blížíící se k počátku
 - Je třeba definovat zarážku, kdy rekurzivní volání zastavit

```
public int Factorial(int n) {  
    if(n == 0 || n == 1) return 1;  
  
    return n * Factorial(n-1);  
}  
  
public int Fibbonaci(int index) {  
  
    if(index == 0 || index == 1) return 1;  
  
    return Fibbonaci(index-1) + Fibbonacci(index-2);  
}
```