Logické operace

Datový typ `bool`

- může nabýt hodnot:
  - `true` → „pravda“, 1, ANO
  - `false` → „nepravda“, 0, NE

Relační operátory

- hodnoty všech primitivních datových typů (`int`, `double`…) jsou uspořádané – lze je porovnávat
- binární (relační) operátory

`>`	větší
`<`	menší
`>=`	větší nebo rovno
`<=`	menší nebo rovno
`==`	rovno
`!=`	nerovno

- Příklad (předpoklad: `int a, b;`):

```
a > b
```

= BV

  - BV (booleovský, logický výraz) = výraz („podmínka“); výsledek → hodnota typu `bool`
- **Nikdy neprovádět test na rovnost dvou reálných čísel** (důvod = zaokrouhlování)

Logické operátory

- `&&` logický součin (AND)
- `||` logický součin (OR)
- `!` negace (NOT)
- **pozor!!!** → `&` a `|` → zcela jiný význam (bitové operace)
- pravdivostní tabulka

`x`	`y`	`x && y`	`x || y`	`!x`
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

- Příklad: Stanovte BV pro test, zda se hodnota proměnné `x` nachází v intervalu (-5; 10⟩ (`true` = nachází, `false` = nenachází)

```
(x > -5) && (x <= 10)
!((x <= -5) || (x > 10))
```

- priorita operátorů → zatím důsledně závorkovat

Řídící struktury

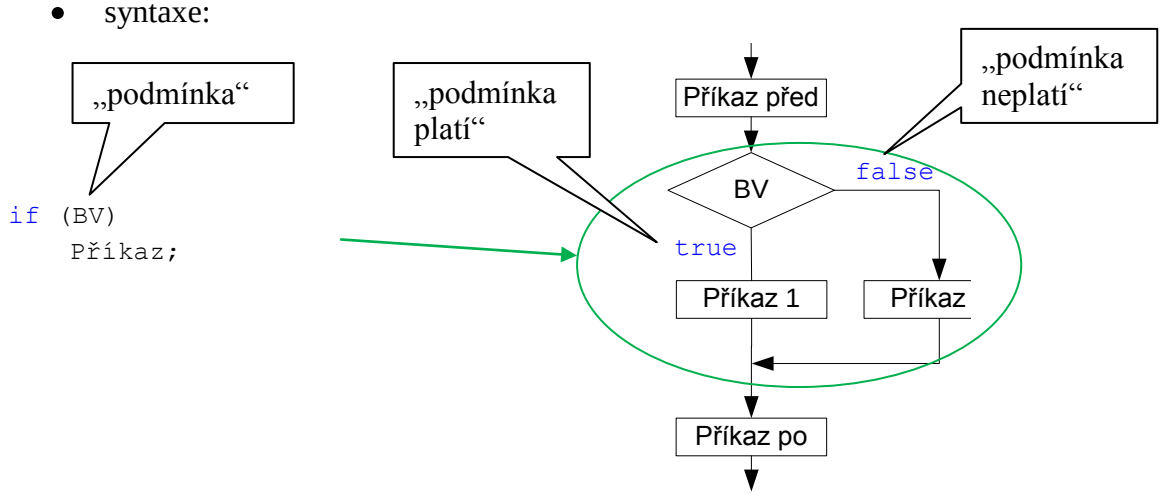
- předepisují způsob provedení jiných příkazů (pořadí, opakování, výběr)
- Řídící struktury:
  - složený příkaz
  - větvení = provedení různých částí programu v závislosti na splnění podmínky
  - cykly = opakování skupiny příkazů

Větvení

- neúplná podmínka `if`
- úplná podmínka `if else`
- vícenásobné větvení `switch`

Neúplná podmínka

- syntaxe:



- sémantika → jestliže je výsledkem `BV` `true`, pak proveď `Příkaz`, jinak jej přeskoč
  - vykonané příkazy (vývojový diagram výše) – podmínka:

platí	neplatí

`Příkaz před`
`Příkaz`
`Příkaz po`
	`Příkaz před`
	`Příkaz po`

- více příkazů uvnitř `if`

Začátek bloku

```
if (BV)
{
    prikaz 1;
    prikaz 2;

    prikaz N;
}
```

konec bloku

`{ }` psát vždy

- Příklad: Převod čísla na absolutní hodnotu, při změně znaménka vytiskne informační zprávu

```
double cislo;
// nacteni z klavesnice
if (cislo < 0)
{
    Console.WriteLine("Menim znamenko");
    cislo = -cislo;
}
Console.WriteLine("|cislo| = {0}", cislo);
```

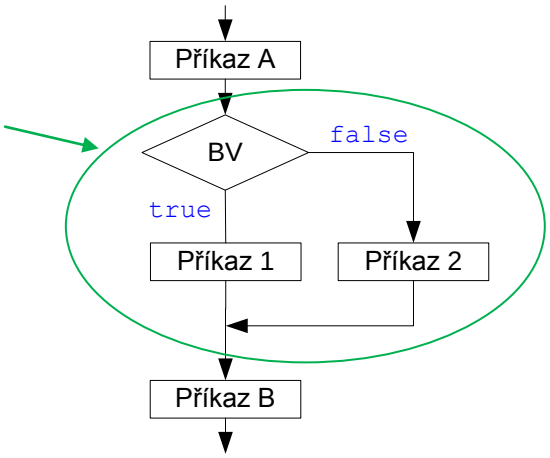
- Poznámka: v BV místo proměnných lze i výrazy.

```
if (10*Math.Sin(uhel) > 0.356)
```

Úplná podmínka

- Syntaxe:

```
if (BV)
    Příkaz 1;
else
    Příkaz 2;
```



- sémantika: jestliže je výsledkem `BV` `true`, pak proveď `Příkaz 1`, jinak proveď `Příkaz 2`
- Příklad: Zachycení dělení nulou

```
double citatel, jmenovatel;
// citatel, jmenovatel - nacteni z klavesnice
double podil = 0;
if (jmenovatel == 0)
{
      Console.WriteLine("Pokus o deleni nulou!!!");
}
else
{
      podil = citatel / jmenovatel;
}
```

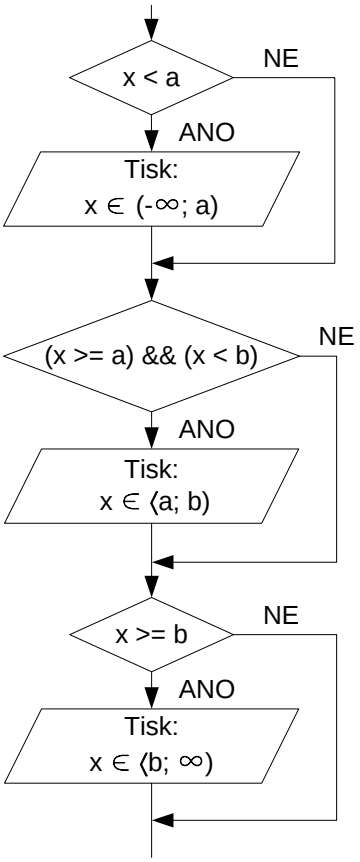
  - Podmínku lze otočit

```
if (jmenovatel != 0)
{
      podil = citatel / jmenovatel;
}
else
{
      Console.WriteLine("Pokus o deleni nulou!!!");
}
```

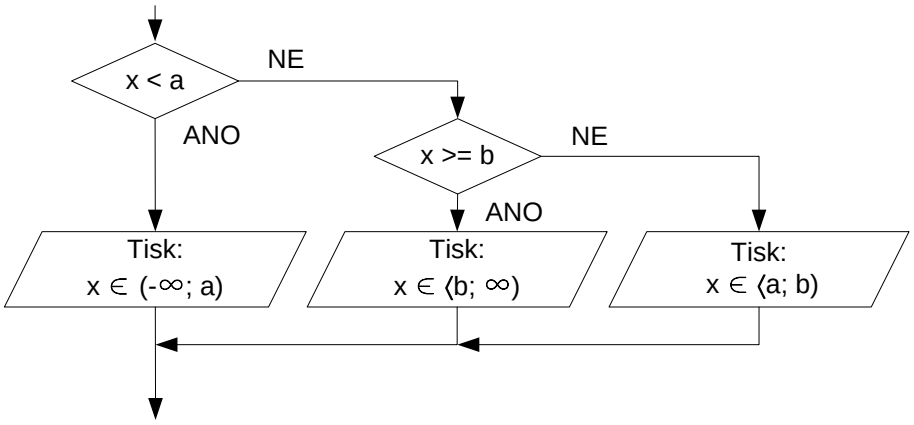
Vícenásobné větvení pomocí `if else`

- Příklad: zjistěte, do kterého intervalu patří x


```
if (x < a)
{
    Console.WriteLine("(-inf; a)");
}
if ((x >= a) && (x < b))
{
    Console.WriteLine("<a; b)");
}
if (x >= b)
{
    Console.WriteLine("<b; inf)");
}
```



- lepší řešení




```
if (x < a)
{
	Console.WriteLine("(-inf; a)");
}
else
{
	if (x >= b)
	{
		Console.WriteLine("<b; inf)");
	}
	else
	{
		Console.WriteLine("<a; b)");
	}
}
```

- Přehlednější zápis → přednostně

```
if (x < a)
{
	Console.WriteLine("(-inf; a)");
}
else if (x >= b)  // if-else = jeden prikaz!!!
{
	Console.WriteLine("<b; inf)");
}
else
{
	Console.WriteLine("<a; b)");
}
```

Složený příkaz (blok)

- = posloupnost příkazů v `{}`
- syntaxe:

```
{
        prikaz 1;
        prikaz 2;

        prikaz N;
}
```

žádný `;`

- použití
  1) sdružení několika příkazů → přehlednost
  2) situace si žádá více příkazů (cykly, podmínky), ale C# vyžaduje příkaz jediný

Viditelnost proměnných

- C# → proměnné viditelné
  - v bloku, kde byly deklarovány
  - do něj vnořených blocích

```
static void Main(string[] args)
{
        int a;
        // kod, lze pouzit: a
        double b;
        // kod, lze pouzit: a,b
        {
                // kod, lze pouzit: a,b
                double c;
                // kod, lze pouzit: a,b,c
        }
        // kod, lze pouzit: a,b
        bool d;
        // kod, lze pouzit: a,b,d
}
```

Cykly

- pro opakované provádění částí programu

Terminologie

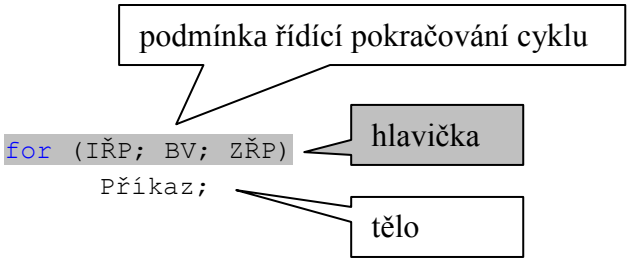
- *řídící proměnná cyklu* = (ŘP) proměnná, na které závisí ukončení cyklu
  - nejlépe pouze jedna
  - jména řídících proměnných → `i`, `j`, `k` …
- *podmínka řídící pokračování cyklu* = logický výraz obsahující řídící proměnnou cyklu
- *hlavička cyklu* = klíčové slovo `for` nebo `while` a výraz v následujících `()`
  - = nutná administrativa cyklu
- *tělo cyklu* = příkazy, které se budou opakovat (výkonný kód cyklu)
  - jeden příkaz nebo blok

Poznámky

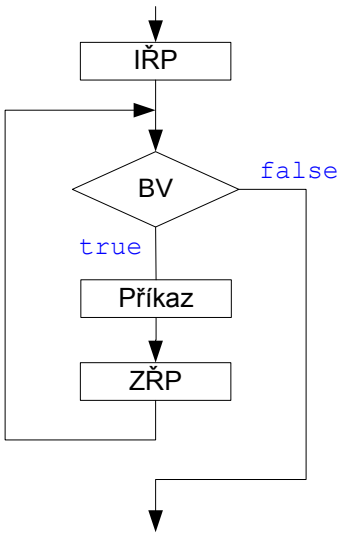
  - podobně jako u `if`, vždy se opakuje pouze jeden příkaz, jinak nutné `{}`

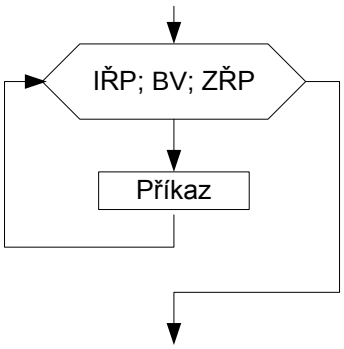
Cyklus for

- pro situace, kdy je počet opakování předem znám
- syntaxe:



- sémantika





Příklady použití

- Součet několika čísel, počet čísel znám již v době zápisu programu

konstanta

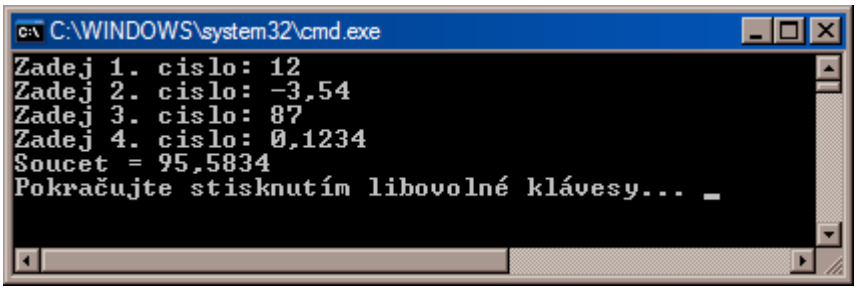
ŘP využita i pro „užitečnou práci“

tělo

postupná akumulace

```
const int POCET = 4;
double cislo, soucet = 0.0;
for (int i = 1; i <= POCET; i++)
{
      Console.Write("Zadej {0}.cislo: ", i);
      cislo = Double.Parse(Console.ReadLine());
      soucet += cislo;
}
Console.WriteLine("Soucet = {0}", soucet);
```

  - viditelnost `i`: hlavička + tělo
  - výsledek.



- Počet čísel není předem znám, zadává se po spuštění programu

`pocet` neznámý

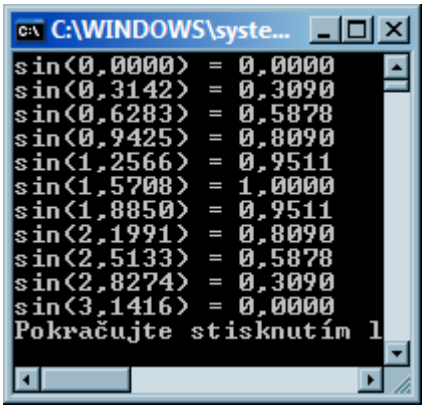
`pocet` znám!!!

```
int pocet;
Console.Write("Zadej pocet cisel: ");
pocet = Int32.Parse(Console.ReadLine());
double cislo, soucet = 0.0;
for (int i = 0; i < pocet; i++)
{
      Console.Write("Zadej {0}. cislo: ", i+1);
      cislo = Double.Parse(Console.ReadLine());
      soucet += cislo;
}
Console.WriteLine("Soucet = {0}", soucet);
```

  - ŘP od 0 → běžnější

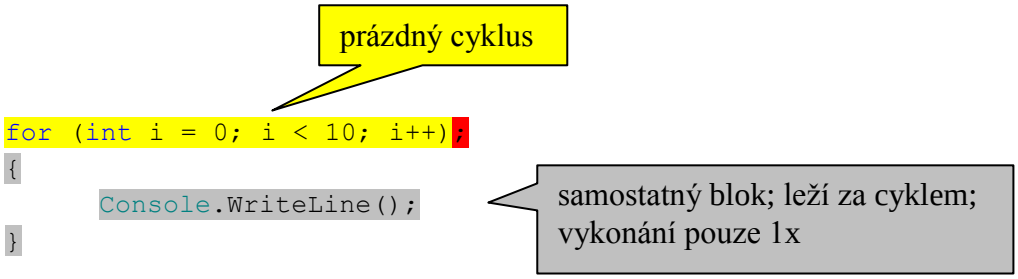
- Méně obvyklé použití – tisk tabulky hodnot funkce sinus

```
for (double x = 0; x <= Math.PI; x += Math.PI / 10)
{
      Console.WriteLine("sin({0:F4}) = {1:F4}", x, Math.Sin(x));
}
```



Poznámky

- Cyklus `for` lze zapsat mnoha dalšími způsoby – nepoužívat je (viz. literatura – např. řídící proměnnou by šlo měnit i uvnitř cyklu)
- pozor na středník za `for`



  - Překladač → varování