C#

KURZ PROGRAMOVÁNÍ

podklady pro 2. část

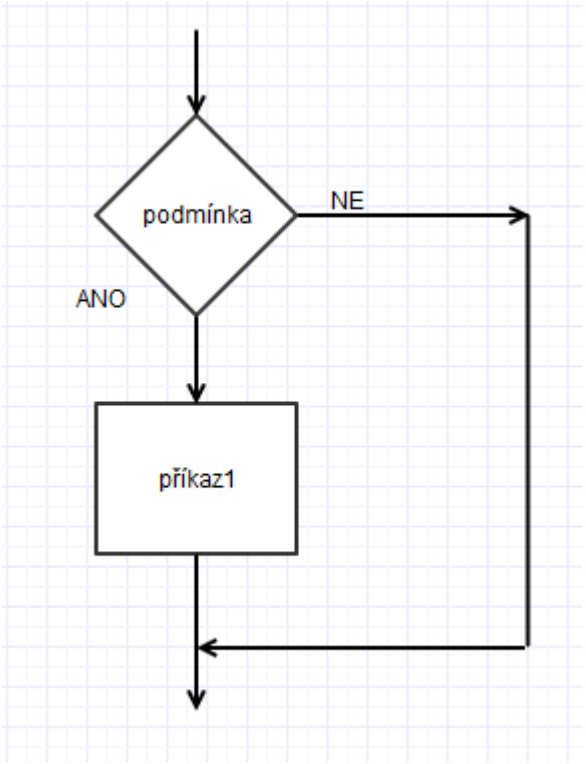
Podmíněné příkazy

- větvení programu
  - vykoná se některá z větví programu
  - potom se dál vykonává následující společná část programu
- if
  - větvení na základě podmínky
- switch
  - přepínač
  - větvení na základě hodnoty

Neúplný podmíněný příkaz if

```
if (podmínka)
        příkaz1;
```

- je-li podmínka splněna, provede se příkaz1;
- není-li podmínka splněna, neprovede se nic



Neúplný podmíněný příkaz - příklady

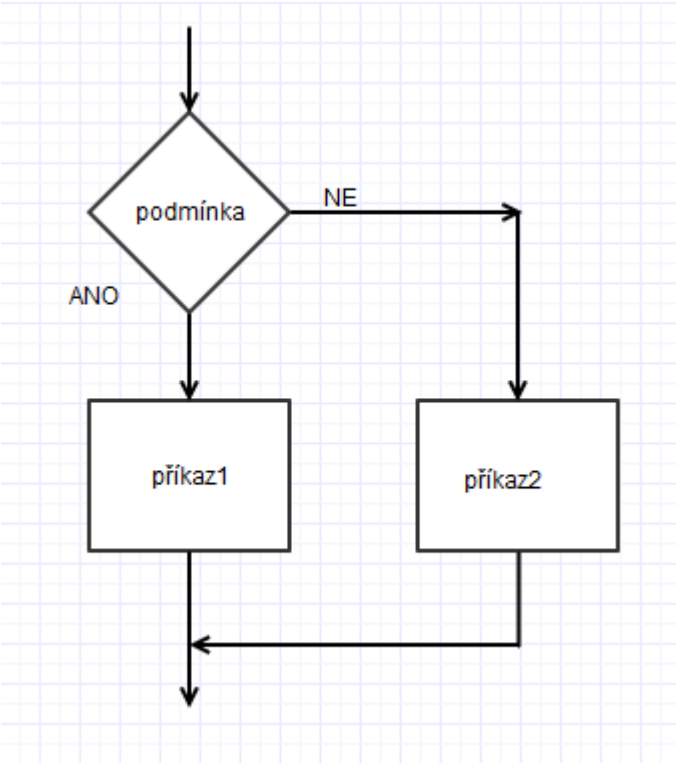
```
if (delitel != 0)
        Console.WriteLine("Podíl je " + (delenec / delitel));
```

- Je-li za podmínkou více než 1 příkaz, použijeme složené závorky (blok příkazů):

```
if (delitel != 0)
{
        double podil = Math.Round(delenec / delitel, 2);
        Console.WriteLine("Podíl je " + podil);
}
```

Úplný podmíněný příkaz if

```
if (podmínka)
        příkaz1;
else
        příkaz2;
```



- je-li podmínka splněna, provede se příkaz1
- v opačném případě (není-li podmínka splněna), se provede příkaz2

Úplný podmíněný příkaz - příklad

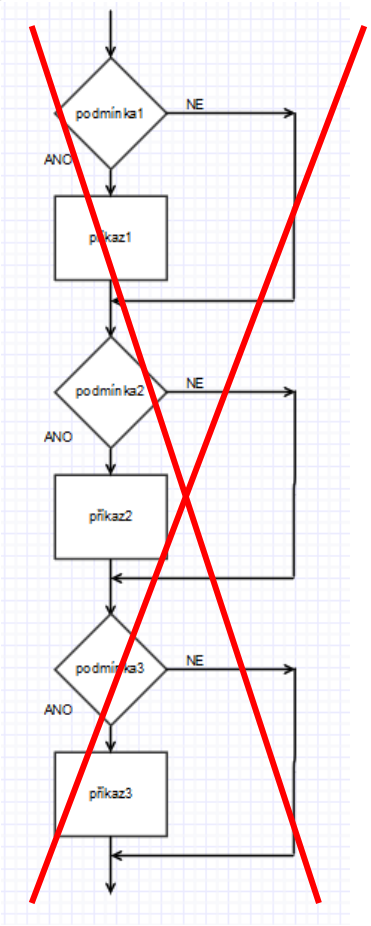
```
if (delitel != 0)
{
        double podil = delenec / delitel;
        Console.WriteLine("Podíl je " + podil);
}
else
      Console.WriteLine("Nulou nemůžeme dělit");
```

- takto raději ne (2 samostatné neúplné podmíněné příkazy):

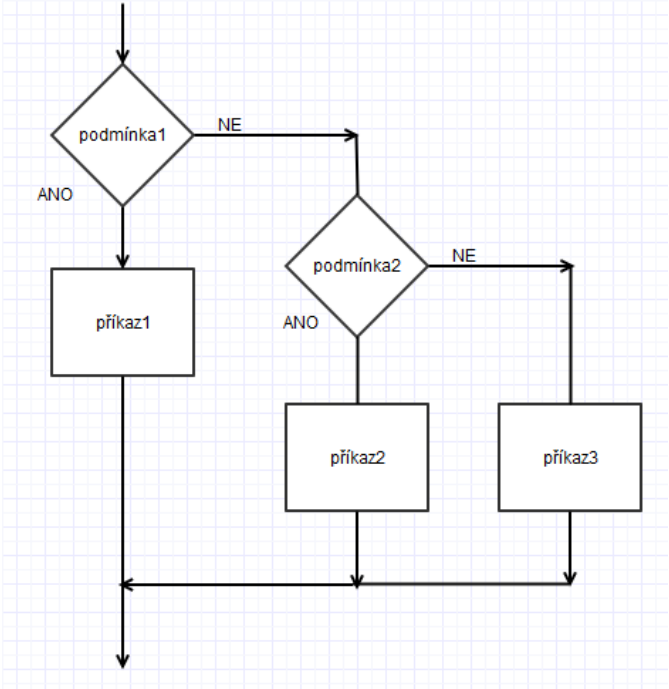
```
if (delitel != 0)
{
        double podil = delenec / delitel;
        Console.WriteLine("Podíl je " + podil);
}
if (delitel == 0)
        Console.WriteLine("Nulou nemůžeme dělit");
```

Vícenásobné větvení programu

- nepoužívejme za sebou řazené neúplné if



- raději využijme několik do sebe vnořených úplných if
  - o 1 příkaz if méně, než je větví (poslední větev je za posledním else)



Přepínač switch

```
switch (proměnná nebo výraz)
{
        case hodnota1:
                prikaz1;
                …
                break;
        …
        case hodnotan:
                prikazn;
                …
                break;
        default:
                prikazx;
                …
                break;
}
```

- jak to funguje:
  - vyhodnotí se *proměnná* nebo *výraz* v závorce
  - podle této hodnoty se provede jedna z větví (mezi *case* a *break*)
  - neodpovídá-li hodnota v závorce žádné z hodnot ve větvích *case*, provede se to, co je za *default* (pokud tam je) nebo nic (*default* je nepovinné)
- bloky příkazů v jednotlivých větvích nemusí být ve složených závorkách (nahrazuje je *break*)
- takto lze větvit na základě konkrétních hodnot, ne obecných podmínek (např. ne podle <0, ==0, >0)

Cyklus

- opakování části programu
  - buď je znám předem počet opakování
  - nebo počet opakování určuje podmínka, která je vyhodnocena až za běhu programu

- druhy cyklů
  - for
    - většinou předem známý počet opakování
  - while
    - cyklus s podmínkou na začátku nebo na konci
  - foreach
    - pro všechny prvky pole (probereme později…)

Cyklus for

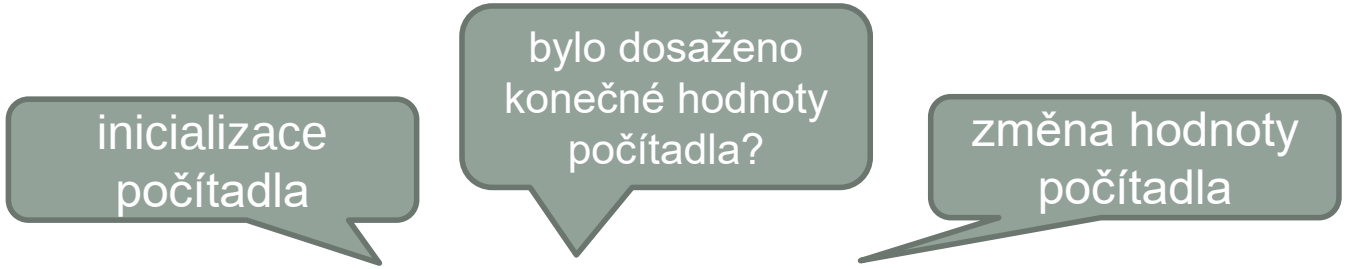
- syntaxe:

```
for (příkaz1; podmínka; příkaz2)
    příkazy_v_těle_cyklu;
```

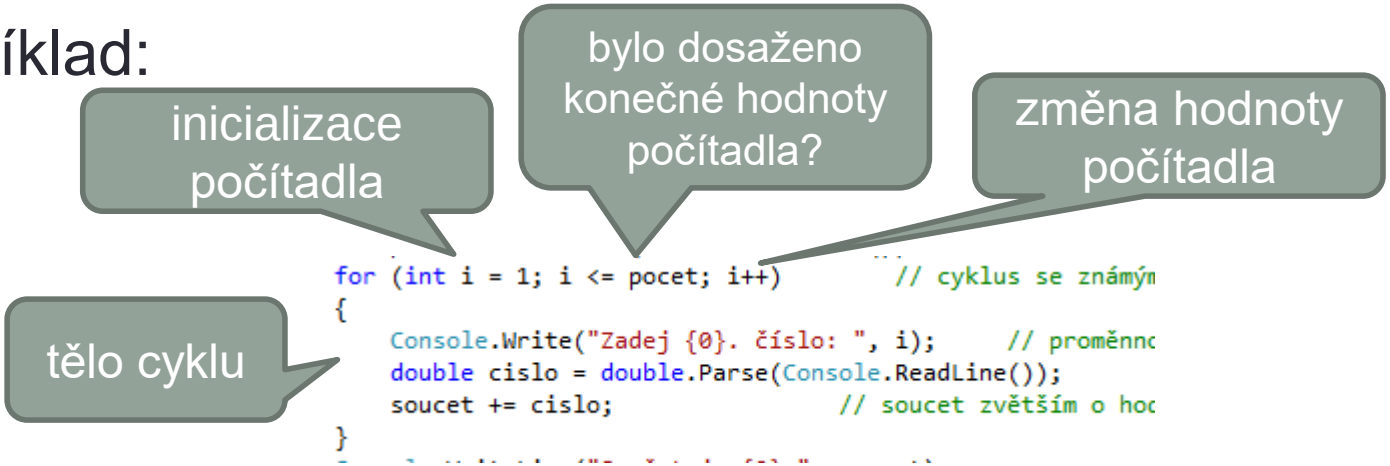
- jak to funguje:
  - *příkaz1*
    - jeden nebo více příkazů (oddělených čárkou), které se provedou **před prvním vstupem do cyklu**
  - *podmínka*
    - vyhodnotí se **před každým provedením těla cyklu**
    - tělo cyklu se provede jen tehdy, je-li podmínka splněna
  - *příkaz2*
    - jeden nebo více příkazů (oddělených čárkou), které se provedou **na konci těla cyklu**
  - *příkazy v těle cyklu*
    - je-li jich více, jsou uvnitř složených závorek { }

Cyklus for
s předem známým počtem opakování

- použijeme pomocnou proměnnou jako počítadlo cyklu
- syntaxe:



- příklad:



Cyklus for

- je-li proměnná (počítadlo) deklarována v hlavičce cyklu, pak existuje jen uvnitř cyklu
  - mimo cyklus ji nelze použít
  - je to **bezpečné**
- potřebujeme-li tuto proměnnou použít i po ukončení cyklu, musíme ji deklarovat ještě před cyklem
- pozor na „ruční“ změnu hodnoty této proměnné v těle cyklu
  - **nebezpečné!**
  - můžeme vyrobit **nekonečný cyklus**

```
for (int i = 1; i <= pocet; i++)        // cyklus se známým počt
{
    Console.Write("Zadej {0}. číslo: ", i);     // proměnnou i m
    double cislo = double.Parse(Console.ReadLine());
    soucet += cislo;                // soucet zvětším o hodnotu
    i = 1;          // toto způsobí, že cyklus nikdy neskončí!
}
```

Cyklus s podmínkou - while

- s podmínkou na začátku:

```
while (podmínka)
        příkazy_v_těle_cyklu;
```

- jak to funguje:
  - *podmínka*
    - vyhodnotí se **před každým provedením těla cyklu**
    - tělo cyklu se provede jen tehdy, je-li podmínka splněna
  - *příkazy v těle cyklu*
    - je-li jich více, jsou uvnitř složených závorek { }

- s podmínkou na konci:

```
do
{
        příkazy_v_těle_cyklu;
}
while (podmínka)
```

- jak to funguje:
  - *příkazy v těle cyklu*
    - vždy se alespoň jednou provedou
  - *podmínka*
    - vyhodnotí se vždy **na konci těla cyklu**
    - tělo cyklu se znovu provede jen tehdy, je-li podmínka splněna

Příklady cyklu while

- s podmínkou na začátku
  - nemusí proběhnout ani jednou
  - (např. je-li hned první číslo 0)

```
while (cislo != 0)                  // dokud číslo není 0...
{
    soucet += cislo;                    // ... přičtu číslo k součtu..
    Console.WriteLine("... přičítám...");                   // pouze "
    Console.Write("Zadávej postupně čísla, zadávání ukonči nulou: ");
    cislo = double.Parse(Console.ReadLine());
}
```

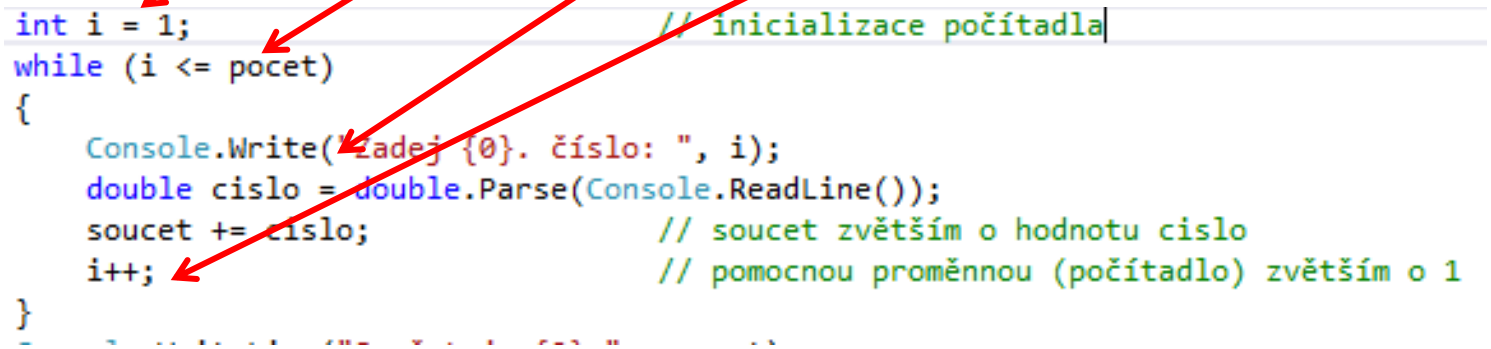
- s podmínkou na konci
  - vždy proběhne alespoň jednou

```
do                                  // opakuj...
{
    Console.Write("Zadávej postupně čísla, zadávání ukonči nulou: ");
    cislo = double.Parse(Console.ReadLine());
    soucet += cislo;
    Console.WriteLine("... přičítám...");                   // pouze "k
} while (cislo != 0);               // ... dokud není číslo 0 (tělo cyk
```

Cyklus while jako náhrada for

for (příkaz1; podmínka; příkaz2)
    příkazy_v_těle_cyklu;

příkaz1;
while (podmínka)
{
    příkazy_v_těle_cyklu;
    příkaz2;
}



```
int i = 1;                              // inicializace počítadla
while (i <= pocet)
{
    Console.Write("Zadej {0}. číslo: ", i);
    double cislo = double.Parse(Console.ReadLine());
    soucet += cislo;                    // soucet zvětším o hodnotu cislo
    i++;                                // pomocnou proměnnou (počítadlo) zvětším o 1
}
```

Cyklus for jako náhrada while

while (podmínka)
    příkazy_v_těle_cyklu;

for (;podmínka;)
    příkazy_v_těle_cyklu;

```
for (; cislo != 0;)
{
    soucet += cislo;                    // ... přičtu číslo k součtu...
    Console.Write("Zadávej postupně čísla, zadávání ukonči nulou: ");
    cislo = double.Parse(Console.ReadLine());
}
```

- nebo i takto
  - ale opatrně!

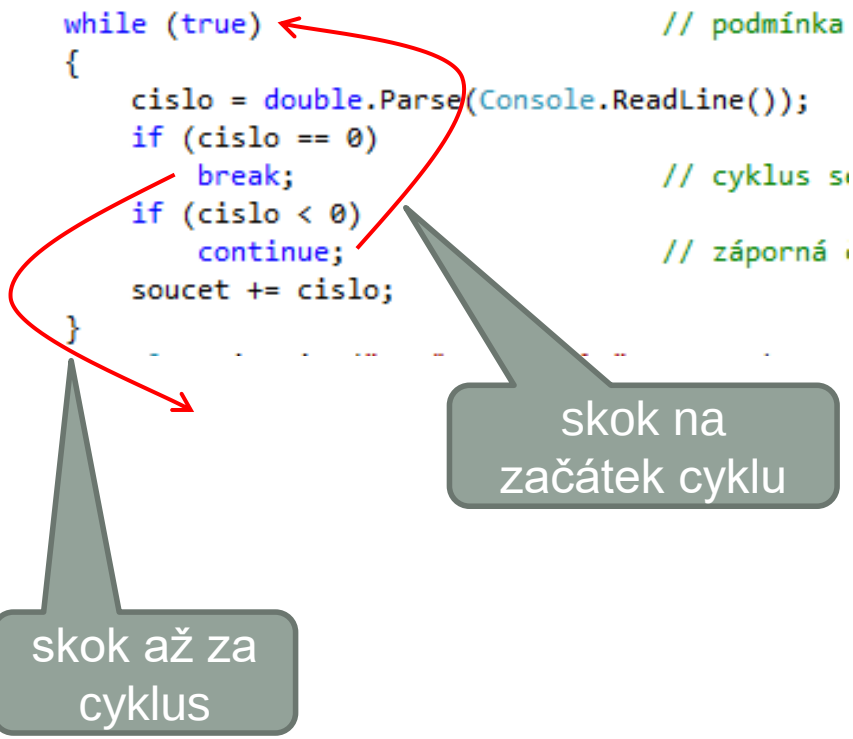
```
for (soucet = 0, cislo = double.Parse(Console.ReadLine()); cislo !=0; soucet += cislo, cislo = double.Parse(Console.ReadLine()))
{
    // tělo cyklu je prázdné
}
```

Příkazy continue a break

- skokové přerušení provádění těla cyklu
  - např. na základě splnění podmínky

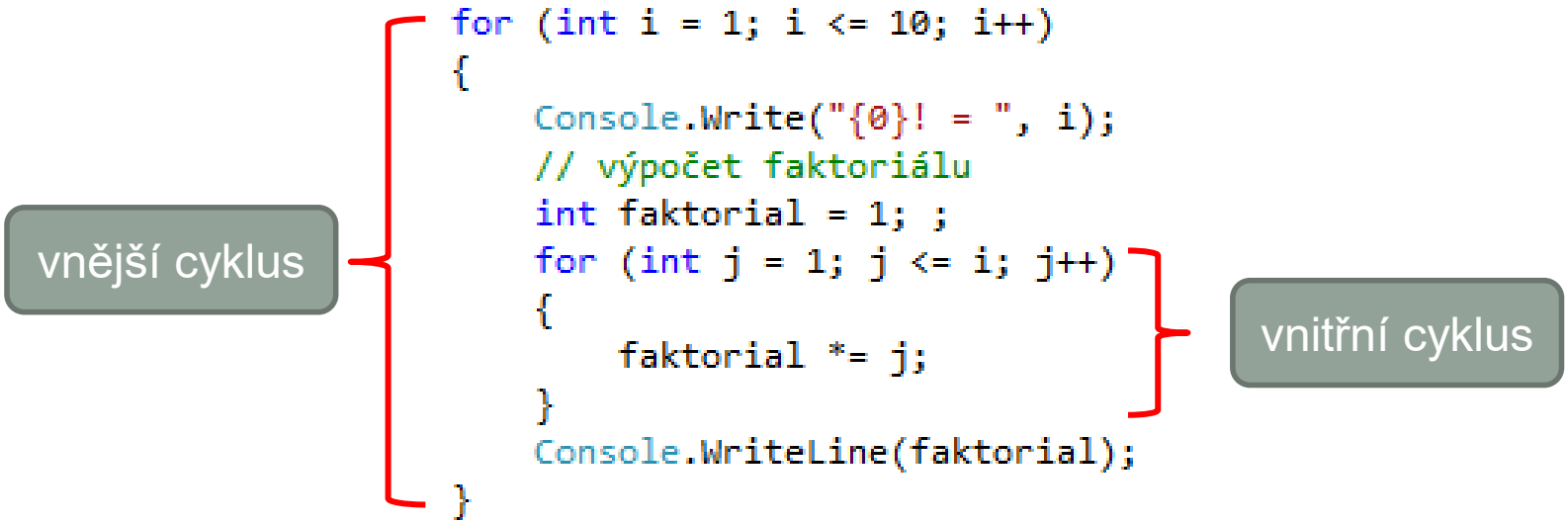
- continue
  - ukončí vykonávání těla cyklu
  - skočí znovu na vstupní podmínku
    - pokud je splněna,
      začne se opět vykonávat tělo cyklu

- break
  - ukončí se vykonávání těla cyklu
  - pokračuje se až za cyklem



Vnořené cykly

- cyklus v cyklu
- používáme-li cyklus for s počítadlem,
  musí být v každém cyklu použita jiná proměnná




```
for (int i = 1; i <= 10; i++)
{
    Console.Write("{0}! = ", i);
    // výpočet faktoriálu
    int faktorial = 1; ;
    for (int j = 1; j <= i; j++)
    {
        faktorial *= j;
    }
    Console.WriteLine(faktorial);
}
```