



Základy programování a algoritmizace Cykly

Erik Král



2020

Informace o autorech:

Ing. et Ing. Erik Král, Ph.D.
Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Nad Stráněmi 4511
760 05 Zlín
ekral@utb.cz



OBSAH

OBSAH			3
		OD	
		CYKLY	
		Řešené příklady	
		ΡΟΙΙΖΊΤΕ Ι ΙΤΕΡΑΤΙΙΚΥ	





1 ÚVOD

V tomto materiálu probereme cykly (loops) [1]. Konrétně probereme příkazy *goto*, *while*, *do-while* a *for*. A dále ukončení cyklu pomocí příkazu break a přeskočení zbytku cyklu pomocí příkazu *continue*.

1.1 Cykly

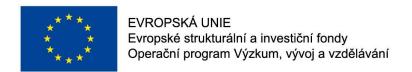
Nejprve probereme příkaz *goto*, který se běžně nepoužívá. Vyjímkou je například ukončení dvou vnořených cyklů *for*. Příkaz goto se většinou nepoužívá proto, že jiné příkazy pro opakování kódu jsou přehlednější.

S pomocí příkazu *goto* můžeme přeskočit na libovolný řádek označený identifikátorem (návěští, anglicky label). V následujícím kódu program podmíněně skočí na řádek označený identifikátorem label.

```
int i = 0;
label:
   Console.WriteLine(i);
   ++i;
   if (i < 10) goto label;</pre>
```

S použitím příkazu *while* můžeme předchozí kód zpřehlednit. Cyklus *while* opakuje příkaz nebo blok příkazů tak dlouho, dokud podmínka v kulatých závorkách vrací true. Cyklus *while* se používá většinou pokud neznáme předem počet opakování.

```
int i = 0;
while (i < 10)
{
    Console.WriteLine(i);
    ++i;
}</pre>
```







Dalším příkazem je příkaz *do-while*. Příkaz *do-while* používáme, pokud nevíme předem počet opakování a chceme, aby se cyklus provedl alespoň jednou.

Následující příklad ukončí cyklus až po tom, co uživatel třikrát stiskne mezerník. Příkaz *Console.ReadKey(true).Key* čeká na stisk klávesy a nemusíme přitom zadávate enter.

```
int n = 0;
do
{
    if (Console.ReadKey(true).Key == ConsoleKey.Spacebar)
    {
        ++n;
    }
    System.Console.WriteLine(n);
} while (n < 3);</pre>
```

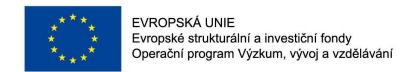
Pokud předem známe počet opakování, tak je nejvhodnější použít cyklus for.

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}</pre>
```

Pojďmě si nyní projít jednotlivé kroky tohoto příkazu.

Jako první se provede definice a inicializace proměnné i:

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}</pre>
```







v dalším kroku se otestuje podmínka i < 10:

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}</pre>
```

pokud podmínka i < 10 vrátí true, tak se provede blok kódu:

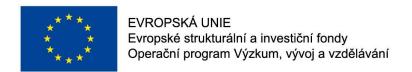
```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}</pre>
```

Po provedení bloku kódu se změní hodnota proměnné i (i++):

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}</pre>
```

A dále se opět poračuje ověřením podmínky i < 10

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}</pre>
```





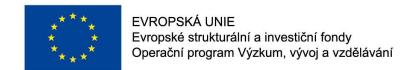


Provádění cyklu můžeme ukončit pomocí příkazu *break*. V následujícím příkazu vypisujeme čísla 0 až 9 po stisknutí libovolné klávesy a výpis ukončíme po stisknutí klávesy *Escape*.

```
for (int i = 0; i < 10; i++)
{
    if(Console.ReadKey(true).Key == ConsoleKey.Escape)
    {
        break;
    }
    Console.WriteLine(i);
}</pre>
```

Zbytek příkazů ve složeném příkazu cyklu můžeme přeskočit pomocí příkazu *continue*. V následujícím příkazu výpis hodnoty na konzoli přeskočíme stiskem klávesy backspace.

```
for (int i = 0; i < 10; i++)
{
    if(Console.ReadKey(true).Key == ConsoleKey.Spacebar)
    {
        continue;
    }
    Console.WriteLine(i);
}</pre>
```





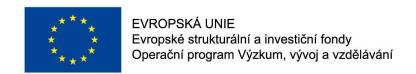


1.2 Řešené příklady

Nyní si ukážeme příklady s kompletním kódem a případně okomentujeme jednotlivá řešení.

První příklad demonstruje využití příkazu goto. V tomto případě není použití příkazu goto vhodné a bylo by vhodnější použít příkaz while. Program vyžaduje od uživatele stiknutí klávesy q.

```
using System;
namespace Test
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Stiskni klavesu q");
        znovu:
            char znak = Console.ReadKey().KeyChar;
            if(znak != 'q')
                Console.WriteLine();
                Console.WriteLine("Mas zadat q");
                goto znovu;
            }
            Console.WriteLine("zadal jsi q, vyborne");
            Console.ReadKey();
        }
    }
```

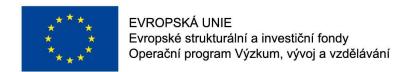






Druhý příklad řeší stejný problém jako předchozí, ale místo příkazu *goto* využívá příkaz *do-while*. Kód je přehlednější než s využitím příkazu *goto*.

```
using System;
namespace Test
{
    class Program
        static void Main(string[] args)
            Console.WriteLine("Stiskni klavesu q");
            char znak;
            do
            {
                znak = Console.ReadKey().KeyChar;
                if(znak != 'q')
                {
                    Console.WriteLine();
                    Console.WriteLine("Mas zadat q");
                }
            }
            while (znak != 'q');
            Console.WriteLine("zadal jsi q, vyborne");
            Console.ReadKey();
        }
    }
```





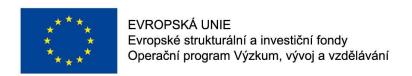


Třetí příklad řeší stejný problém jako předchozí příklad, ale tentokrát s využitím cyklu *while*. Všimněte si, jak je kód kratší než předchozí řešení.

```
using System;
namespace Test
{
    class Program
        static void Main(string[] args)
            Console.WriteLine("Stiskni klavesu q");
            char znak;
            while ((znak = Console.ReadKey().KeyChar) != 'q')
            {
                 Console.WriteLine();
                 Console.WriteLine("Mas zadat q");
            }
            Console.WriteLine("zadal jsi q, vyborne");
            Console.ReadKey();
        }
    }
```

Pozornost si zaslouží zápis podmínky, který využívá toho, že operátor přiřazení vrací přiřazenou hodnotu. Nejprve se tedy provede operace přiřazení $z_{nak} = C_{onsole.Read-Key().KeyChar}$ která vrátí hodnotu přiřazeného znaku a tato hodnota se potom porovná se znakem q.

```
(znak = Console.ReadKey().KeyChar) != 'q'
```



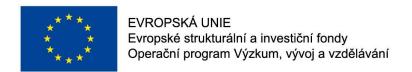




V následujících kódech si postupně projdeme několik příkladů na příkaz for.

První příklad na cyklus for vypíše na konzolí čísla cisla 1,2,3,4,5,6,7,8,9.

Druhý příklad na cyklus *for* vypíše čísla 10, 9, 8, 7, 6, 5, 4, 3, 2, 1. Všimněte si snižování hodnoty i–







Třetí příklad na cyklus for vypíše čísla 10,20,30,40,50,60,70,80,90,100. Všimněte si změny hodnoty proměnné i = i + 10 a podmínky i <= 100.

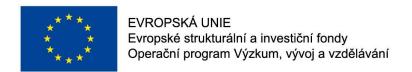
```
using System;

namespace ConsoleApp6
{
    class Program
    {
        static void Main(string[] args)
         {
            for (int i = 10; i <= 100; i = i + 10)
            {
                 Console.WriteLine(i);
            }
          }
        }
}</pre>
```

Čtvrtý příklad na cyklus *for* vypíše čísla 10,100,1000,10000,100000. Tentokrát měníme hodnoty pomocí výrazu i = i * 10.

```
using System;

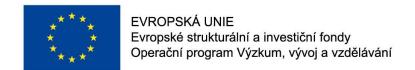
namespace ConsoleApp6
{
    class Program
    {
        static void Main(string[] args)
         {
            for (int i = 10; i <= 100000; i = i * 10)
            {
                 Console.WriteLine(i);
            }
          }
    }
}</pre>
```







Poslední příklad na cyklus *for* vypíše čísla 256, 128, 64, 32, 16, 8, 4, 2, 1. Proměnná i je inicializovaná na hodnotu 256 a potom je v každé iterace vydělena dvěma. Všimněte si podmínky i >= 1.



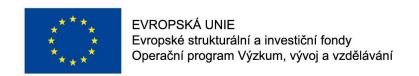




Další **příklad na cyklus** *for* je příklad na výpočet faktoriálu. Na programu je nejdůležitější, že si nadefinujeme proměnnou f, kterou potom v iteracích násobíme postupně snižovanou hodnotou n.

Všimněte si, že v cyklu for může být vynechaná definice iterační proměnné.

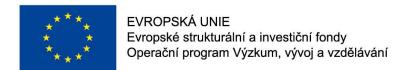
```
for (; n > 1; n--)
```







Poslední příklad, který je zde uvedený vypíše vypíše čísla od 0 do 100 dělitelná 5 nebo 3. Pro to využívá modulo operátor [2] i % 5 a i % 3 což je zbytek po celočíselném dělení.







SEZNAM POUŽITÉ LITERATURY

- [1] Branches and loops Introduction to C# interactive tutorial | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 04.01.2021]. Dostupné z: https://docs.microsoft.com/sk-sk/dotnet/csharp/tutorials/intro-to-csharp/branches-and-loops?tutorial-step=3
- [2] Arithmetic operators C# reference | Microsoft Docs. [online]. Copyright © Microsoft 2021 [cit. 04.01.2021]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators#remain-der-operator-

