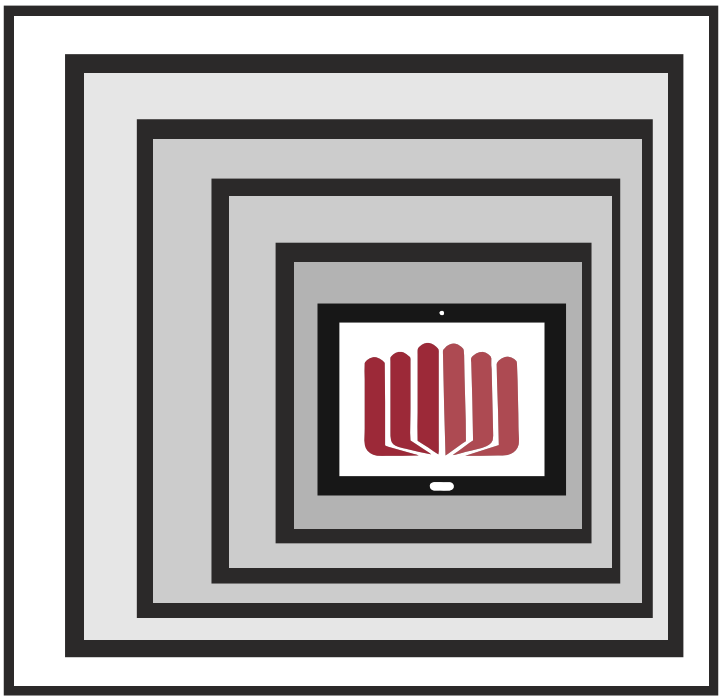
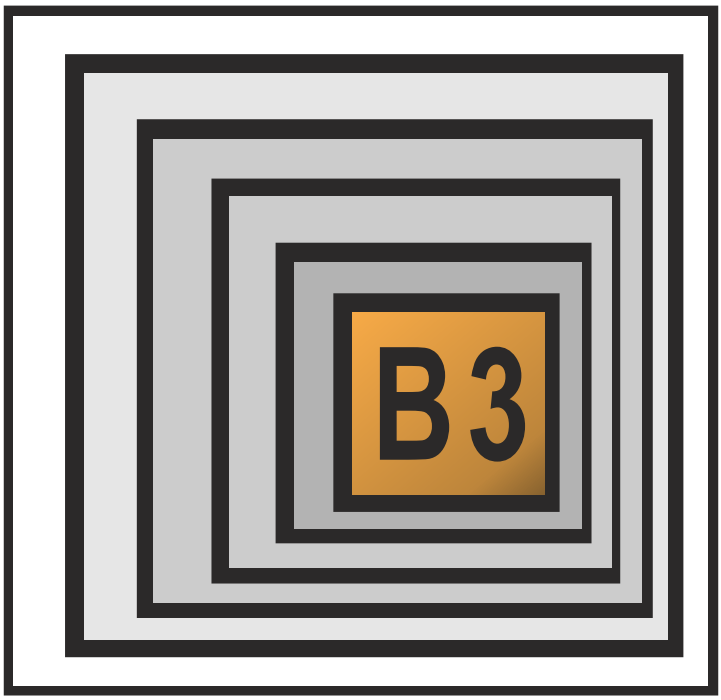
evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ



KONZOLOVÉ APLIKACE V C#



KONZOLOVÉ APLIKACE V C#

RNDr. Jan Preclík, Ph.D.

27. 07. 2015

OBSAH

Jednoduché konzolové aplikace v C#

Cíl: Procvičení si syntaxe jazyka C# na jednoduchých konzolových aplikacích.

Požadavky: Vývojové prostředí pro jazyk C#, například Open Source prostředí SharpDevelop (http://www.icsharpcode.net/opensource/sd).
Nainstalované potřebné knihovny .NET.
Základní znalost technické angličtiny potřebné pro ovládání prostředí.

Veškeré úlohy budou demonstrovány na vývojovém prostředí SharpDevelop. Lze ale využít i Microsoft Visual Studio Express.

Cvičení jsou umístěna do samostatných složek, každá složka obsahuje:

- soubor **zadání.docx**, (jednotlivá zadání jsou součástí této brožury) zde najdete:
  - stručné zadání úlohy, většinou i včetně ukázky očekávaného výstupu
  - stručný postup, jak úlohu řešit – zde jsou zmíněny důležité body, které je potřeba vyřešit, včetně různých možností řešení a stručného připomenutí potřebné teorie; toto by mělo žákům postačovat k samostatnému vyřešení úlohy (předpokládám, že potřebná teorie už jim byla podrobněji vysvětlena)
  - vzorové řešení – na poslední stránce/stránkách; s výjimkou prvního cvičení je v řešení uváděna jen „důležitá" část kódu – je vynechána část ***using*** (pokud zde není potřeba něco měnit), deklarace jmenného prostoru a třídy ***Program***
- ***.sln** – uložený projekt (řešení) z prostředí ***SharpDevelop*** se vzorovým řešením úlohy.

Pokud se po spuštění vámi vytvořené konzolové aplikace stane, že její okno bude příliš malé a texty v něm špatně čitelné, v menu tohoto okna vyberte **Vlastnosti** a zvolte vhodnou velikost písma.

Cvičení 01 – ahoj lidi

Nedá se začít jinak…
Zde jsou vysvětleny základy ovládání prostředí (našeptávač…) a psaní komentářů.

Cvičení 02 – pozdrav

Deklarace proměnné a vstup od uživatele. Escape sekvence.

Cvičení 03 – papoušek

Vstup od uživatele, výpis na obrazovku a využití zástupných symbolů.

Cvičení 04 – ELIZA

Opakování pomocí cyklu s podmínkou na začátku.

Cvičení 05 – obvod, obsah čtverce

Načítání číselných hodnot od uživatele (parsování vstupu).

Cvičení 06 – obsahy obrazců

Načítání číselných hodnot (parsování vstupu), podmínky (***if, switch***).

Cvičení 07 – lineární rovnice

Načítání číselných hodnot (parsování vstupu), podmínky.

Cvičení 08 – kvadratická rovnice

Matematické výpočty, třída ***Math***.

Cvičení 09 – trojúhelník

Složené podmínky.

Cvičení 10 – výpočet BMI

Vícenásobné podmínky, formátovací řetězce.

Cvičení 11 – součet a průměr

Cyklus s podmínkou na začátku, operátory ++, --, += ...
přetypování (konverze číselných hodnot).

Cvičení 12 – součet a průměr

Stejné jako předchozí, ale s využitím cyklu s podmínkou na konci.

Cvičení 13 – hvězdičky

Cyklus ***for***.

Cvičení 14 – trojúhelník z hvězdiček

Cyklus ***for*** a vnořování cyklů do sebe.

Cvičení 15 – faktoriál

Různé druhy cyklů a jejich porovnání.
Porovnání rozsahu různých číselných datových typů.

Zadání a řešení jednotlivých cvičení

Cvičení 01 – ahoj lidi

Úkol

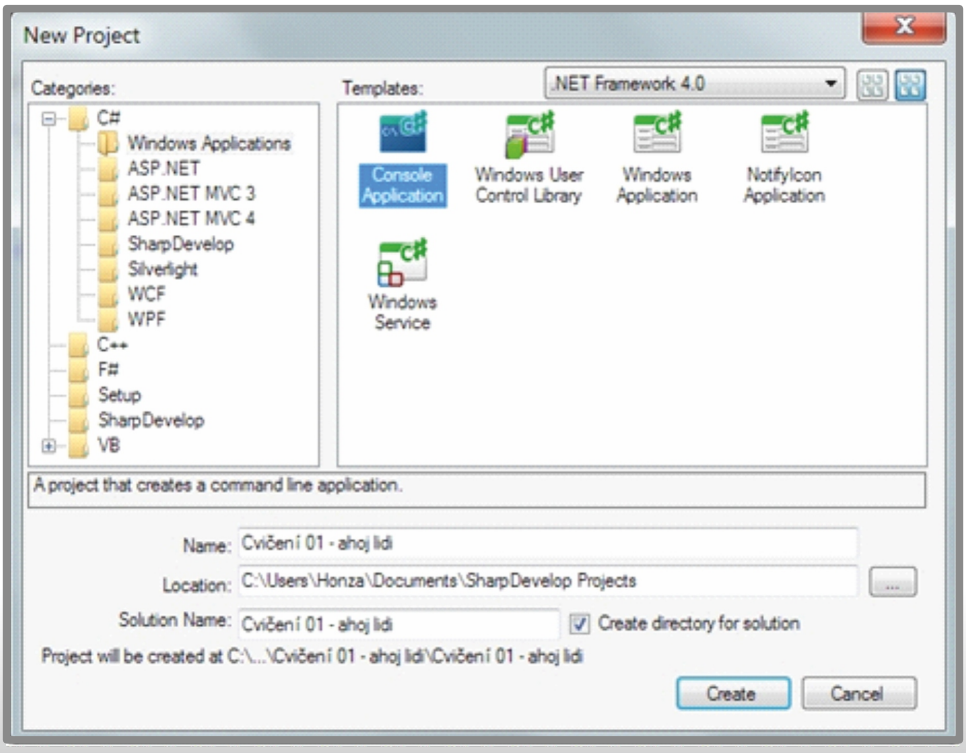
Vytvořte konzolovou aplikaci, která na obrazovku vypíše text „Ahoj lidi!“. Na další řádek vypíše „čekám na stisk klávesy…“ a po stisku libovolné klávesy se ukončí. Každý příkaz okomentujte.

Procvičíte si

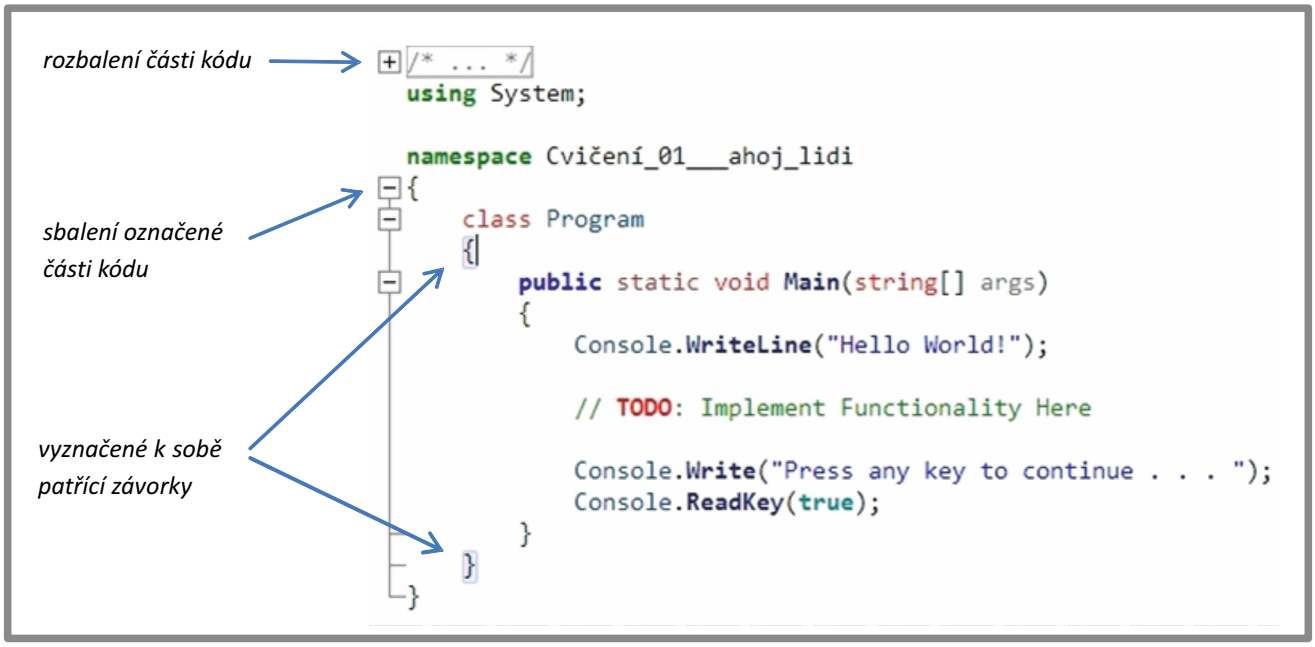
- základy ovládání prostředí ***SharpDevelop*** – založení nového řešení, našeptávač
- výpis textu na obrazovku
- čekání na stisk klávesy
- psaní komentářů

Stručný postup

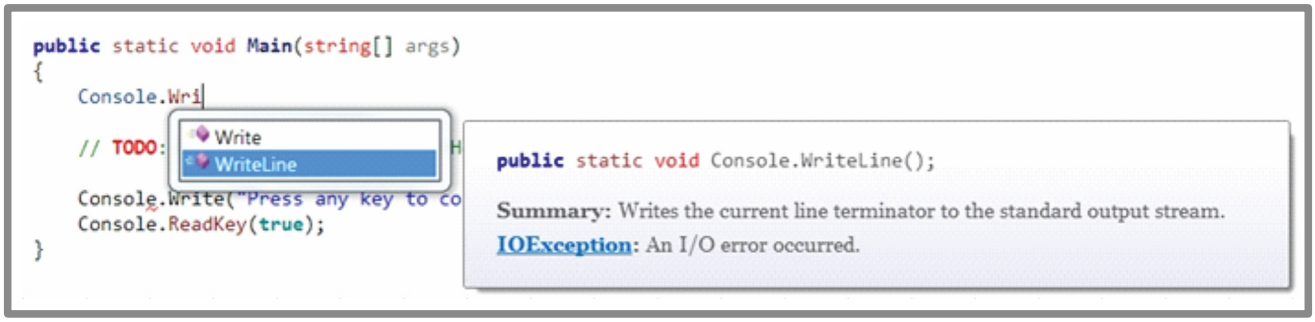
1. Založte nový projekt (= řešení, solution) – menu ***File/New/Solution***…, zvolte ***C#, Windows Applications, Console Application***.
2. Zadejte název tohoto řešení do pole ***Name*** (lze i s diakritikou), vyberte umístění na disku (***Location***). ***Solution Name*** se doplní automaticky, doporučuji neměnit. Pokud ponecháte označené zatržítko ***Create directory for solution***, bude pro toto řešení vytvořena nová složka, doporučuji.



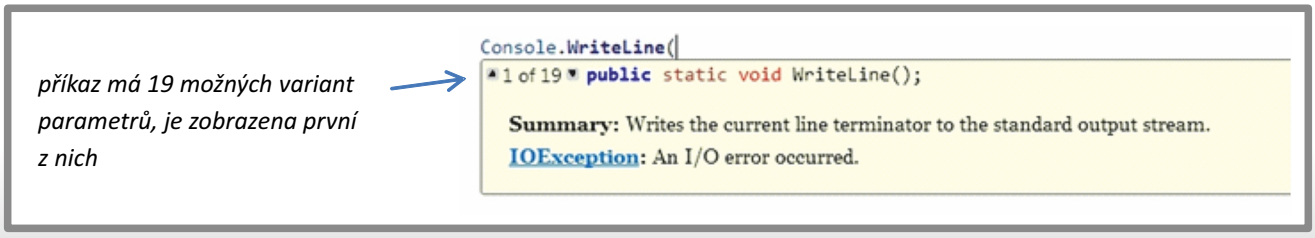
3. Kliknutím na symboly + a – můžete označené části kódu rozbalit a sbalit.
   Využívejte toho ke zpřehlednění vašeho kódu, právě nepotřebné části sbalte, aby zbytečně nezabíraly místo.
   Při umístění kurzoru na složenou závorku se zvýrazní druhá jí odpovídající závorka.



4. Při psaní příkazu „našeptávač“ zobrazuje jeho možná dokončení, požadované můžete vybrat pomocí kurzorových kláves ***nahoru/dolů*** a potvrdit klávesou ***ENTER***.
   Našeptávač můžete také vyvolat pomocí ***CTRL + mezerník***.



5. Našeptávač také zobrazuje parametry právě dokončovaného příkazu, pokud má příkaz více variant parametrů, přecházíte mezi nimi kurzorovými klávesami ***nahoru*** a ***dolů***.



6. Vyzkoušejte si také psaní komentářů:

```
// jednořádkový komentář až do konce řádku

/* víceřádkový komentář
   až do značky */
```

Řešení

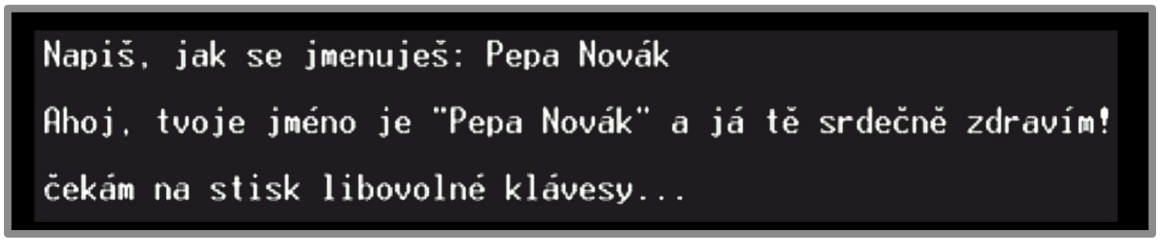
```
using System;

namespace Cvičení_01___ahoj_lidi
{
  class Program
  {
    public static void Main(string[] args)
    {
      Console.WriteLine("Ahoj lidi");  // pozdravíme a odřádkujeme,
                                       // tj. přejdeme na nový řádek, další
                                       // text tak bude vypsán na
                                       // nový řádek
      Console.Write("Čekám na stisk klávesy...");  /* toto je víceřádkový
                                 komentář, porovnejte s předcházejícím
                                 komentářem, který platí jen do konce řádku */

      Console.ReadKey();  // čekáme na stisk libovolné klávesy
    }
  }
}
```

Cvičení 02 – pozdrav

Úkol

Vytvořte konzolovou aplikaci, která nejprve vyzve uživatele k zadání jeho jména a pak ho slušně pozdraví. Ukončí se po stisku libovolné klávesy.

Vypadat to může třeba takto:



Procvičíte si

- výpis textu na obrazovku, odřádkování
- načítání vstupu od uživatele
- escape sekvence

Stručný postup

1. Rozdíl mezi metodami ***Console.Write*** a ***Console.WriteLine*** je v tom, že druhá metoda po výpisu textu odřádkuje, další výpis tak již bude probíhat na nový řádek.

   Odřádkovat lze také pomocí speciálního znaku (***escape sekvence***) **\n**.

   Když tedy chceme vypsat text do dvou řádků, můžeme použít

   ```
   Console.WriteLine("první řádek");
   Console.WriteLine("druhý řádek");
   ```

   nebo

   ```
   Console.WriteLine("první řádek\ndruhý řádek");
   ```

2. Další užitečné ***escape sekvence*** mohou sloužit k vypsání uvozovek (\") nebo apostrofu (\').

   ```
   Console.WriteLine("následuje \"text v uvozovkách\" a
                      \'text v apostrofech\'");
   ```

   Seznam escape sekvencí najdete třeba na ***https://msdn.microsoft.com/en-us/library/h21280bw.aspx***.

3. Proměnnou v C# je možno nejprve deklarovat (tj. oznámit překladači, jak se bude jmenovat a jakého bude typu), a poté jí přiřadit hodnotu

```
string jmeno;
jmeno = Console.ReadLine();
```

nebo to lze udělat v jednom kroku

```
string jmeno = Console.ReadLine();
```

Řešení

```
Console.Write("Napiš, jak se jmenuješ: ");  // odřádkovávat nebudeme, abychom
                                            // mohli svoje jméno psát hned za
                                            // tuto výzvu a ne až na nový
                                            // řádek
string jmeno = Console.ReadLine();  // deklarace proměnné a přiřazení hodnoty
                                    // vstupem od uživatele
Console.WriteLine();  // vynecháme prázdný řádek

Console.WriteLine("Ahoj, tvoje jméno je \"" + jmeno + "\" a já tě srdečně
zdravím!\n");  // ukázka použití escape sekvencí pro vypsání uvozovek
               // a odřádkování

Console.Write("Čekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 03 – papoušek

Úkol

Vytvořte konzolovou aplikaci „simulující papouška“, cokoliv napíšete, aplikace zopakuje 3.

Vypadat to může třeba takto:

```
Já jsem tvůj papoušek, co napíšeš, to zopakuji!
Napiš něco: nazdar
Papoušek říká nazdar, nazdar, nazdar.
čekám na stisk libovolné klávesy...
```

Procvičíte si

- výpis textu na obrazovku, zástupné symboly

Stručný postup

1. Pokud chceme načtené slovo (do proměnné ***slovo***) vypsat 3 za sebou, můžeme použít operátor **+**, který řetězce spojí dohromady (zřetězí). Není to ale příliš přehledné.

```
Console.WriteLine("Papoušek říká "+slovo+", "+slovo+", "+slovo+".");
```

Lepší je využít ***zástupné symboly {0}, {1}…*** Každý zástupný symbol bude při výpisu řetězce nahrazen hodnotou příslušného parametru (*{0}* hodnotou prvního parametru, *{1}* hodnotou druhého parametru…).

```
Console.WriteLine("Papoušek říká {0}, {1}, {2}.", slovo, slovo, slovo);
```

Stačí nám ale jen jeden zástupný symbol a tím i jeden parametr.

```
Console.WriteLine("Papoušek říká {0}, {0}, {0}.", slovo);
```

Řešení

```
Console.WriteLine("Já jsem tvůj papoušek, co napíšeš, to zopakuji!\n");
    // díky \n vynecháme za textem prázdný řádek

Console.Write("Napiš něco: ");
string slovo = Console.ReadLine();  // deklarace proměnné slovo,
                                    // hned načteme vstup od uživatele
Console.WriteLine("\nPapoušek říká {0}, {0}, {0}.\n", slovo);
    // \n na začátku zajistí vynechání prázdného řádku před tímto textem

Console.Write("Čekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 04 – ELIZA

Úkol

Vytvořte konzolovou aplikaci fungující na hodně zjednodušeném principu legendárního programu ELIZA (podívejte se na ***https://cs.wikipedia.org/wiki/ELIZA***).

Nejprve se vás program zeptá, jak vás má oslovovat. A poté na cokoliv, co napíšete, odpoví „Opravdu si myslíte, že …“ a zopakuje to, co jste napsali. Tento postup simuluje jednu z technik, které používají psychoterapeuti.

Aplikace se ukončí po zadání textu „konec“.

Vypadat to může třeba takto:

```
Ahoj, já jsem ELIZA!
Jak Vám mám říkat? Honzo
Honzo, napište mi něco: venku prší
Opravdu si myslíte, že venku prší?

Honzo, napište mi něco: modrá je dobrá
Opravdu si myslíte, že modrá je dobrá?

Honzo, napište mi něco: konec
```

Procvičíte si

- výpis textu na obrazovku, zástupné symboly
- cyklus s podmínkou na začátku

Stručný postup

1. Pro opakování části kódu využijeme cyklus s podmínkou na začátku:

   `while (podmínka) {tělo cyklu}`

   Tento cyklus se provádí, dokud podmínka platí (tj. dokud je splněna).

   Nezapomeňte, že relační (porovnávací) operátory v C# mají tvar:

   `==  !=  <  >  <=  >=`

   Základ našeho programu může vypadat tak, že do proměnné veta budeme načítat vstup od uživatele a to opakovat tak dlouho, až uživatel zadá text konec, potom se cyklus ukončí.

```
string veta = "";
while (veta != "konec") {
  Console.Write("\n{0}, napište mi něco: ", osloveni);
  veta = Console.ReadLine();
  Console.WriteLine("\nOpravdu si myslíte, že {0}?\n", veta);
}
```

Poznámka: Můžete si všimnout, že toto není úplně ideální řešení, protože po zadání textu „konec“ aplikace ještě vypíše „Opravdu si myslíte, že konec?“ a pak se teprve ukončí.

Řešení

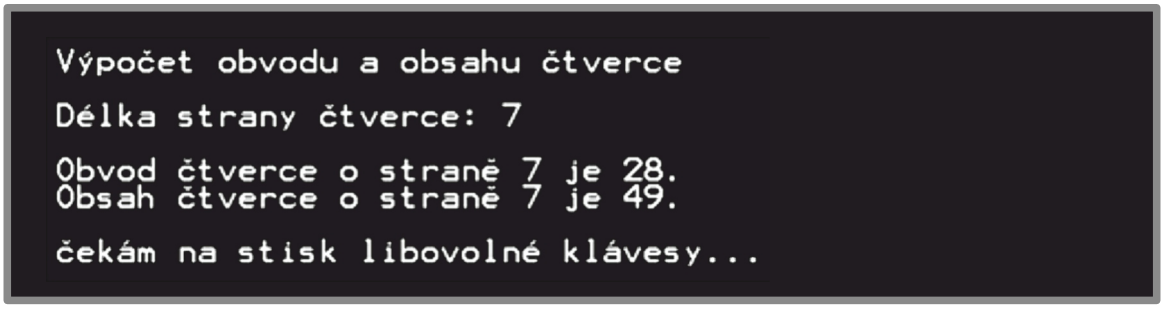
```
Console.WriteLine("Ahoj, já jsem ELIZA!\n");  // \n vynechá za textem
                                              // prázdný řádek

Console.Write("Jak Vám mám říkat? ");
string osloveni = Console.ReadLine();  // deklarace proměnné osloveni,
                                       // hned načteme vstup od uživatele
string veta = "";                      // deklarace proměnné veta,
                                       // přiřadíme jí prázdný řetězec

while (veta != "konec") {      // opakuje se tak dlouho, dokud nebude
                               // zadáno slovo "konec"
  Console.Write("\n{0}, napište mi něco: ", osloveni);
  veta = Console.ReadLine();  // načteme odpověď uživatele

  Console.WriteLine("\nOpravdu si myslíte, že {0}?\n", veta);
}
```

Cvičení 05 – obvod, obsah čtverce

Úkol

Vytvořte konzolovou aplikaci, která pro zadanou délku strany spočítá a vypíše obvod a obsah čtverce.

Vypadat to může třeba takto:



Procvičíte si

- číselné datové typy
- načítání číselných hodnot (parsování vstupu)

Stručný postup

1. Základní číselné datové typy (viz ***https://msdn.microsoft.com/cs-cz/library/s1ax56ch.aspx***), které můžeme v C# využívat, jsou

   - **celočíselné**
     - **se znaménkem** (signed)
       `sbyte` (–128 .. 127)
       `short` (–32 768 .. 32 767)
       `int` (–2 147 483 648 .. 2 147 483 647)
       `long` (–9 223 372 036 854 775 808 .. 9 223 372 036 854 775 807)
     - **bez znaménka** (unsigned)
       `byte` (0 .. 255)
       `ushort` (0 .. 65 535)
       `uint` (0 .. 4 294 967 295)
       `ulong` (0 .. 18 446 744 073 709 551 615)
   - **reálné**

     `float` (–3.4 × 10^{38} .. +3.4 × 10^{38}, přesnost 7 významných číslic)
     `double` (±5.0 × 10^{-324} .. ±1.7 × 10^{308}, přesnost 15–16 významných číslic)
     `decimal` (–7.9 10^{28} .. 7.9 10^{28}, přesnost 28–29 významných číslic)

2. Chceme-li číselnou konstantu označit, že je daného typu, musíme za číslem použít příslušnou příponu (***F*** pro ***float***, ***D*** pro ***double***, ***U*** pro ***uint***, ***L*** pro ***long***, ***UL*** pro ***ulong***, ***M*** pro ***decimal***). Přípony mohou být i malými písmeny, ale pak pozor na záměnu písmene l a číslice 1.

```
float realneCislo1 = 13.5F;

double realneCislo2 = 147.78D;
double realneCislo3 = 147.78;      // lze také bez přípony

uint celeCislo1 = 1234U;
uint celeCislo2 = 1234;            // lze také bez přípony

long dlouheCeleCislo1 = 1234L;
long dlouheCeleCislo2 = 1234;      // lze také bez přípony

ulong dlouheCeleCislo3 = 1234UL;
ulong dlouheCeleCislo4 = 1234;     // lze také bez přípony

decimal mena = 34.56M;
```

3. Textový vstup (datový typ ***string***) již načítat umíme, pro načtení celého nebo reálného čísla od uživatele využijeme toho, že v C# je „všechno objekt“ a objekty se „o sebe umí postarat“. Takže objekt ***int*** nebo ***float*** se umí načíst z textového řetězce pomocí své metody ***Parse()***.

   Můžeme nejprve načíst do proměnné radek vstup od uživatele a pak do proměnné strana hodnotu čísla v tomto textovém řetězci, to uděláme pomocí metody ***Parse*** příslušného objektu (pokud chceme načítat celé číslo, pak ***int.Parse***, pokud reálné, pak ***float.Parse***).

```
string radek = Console.ReadLine();
float strana = float.Parse(radek);
```

   Tyto dva příkazy ale můžeme spojit do jednoho:

```
float strana = float.Parse(Console.ReadLine());
```

4. Při výpisu můžeme nejprve vypočítat výslednou hodnotu, uložit do proměnné a poté vypsat. To je výhodné, pokud s touto výslednou hodnotou budeme ještě dále pracovat.

```
float obvod;
obvod = 4 * strana;
Console.WriteLine("Obvod čtverce je {0}.", obvod);
```

Deklaraci proměnné můžeme spojit s její inicializací:

```
float obvod = 4 * strana;
```

Pokud s výslednou hodnotou už nebudeme dále pracovat, nemusíme pro ni deklarovat novou proměnnou, ale můžeme výpis a výpočet spojit dohromady.

```
Console.WriteLine("Obvod čtverce je {0}.", 4*strana);
```

Řešení

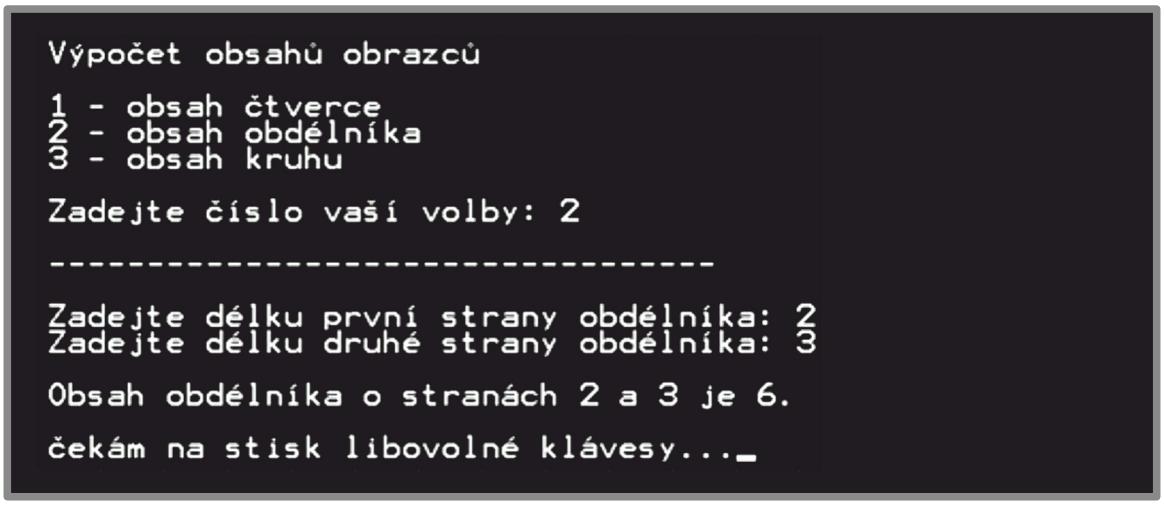
```
Console.WriteLine("Výpočet obvodu a obsahu čtverce\n");

Console.Write("Délka strany čtverce: ");
float strana = float.Parse(Console.ReadLine());
    // načtení textového vstupu od uživatele
    // a jeho převod na reálné číslo pomocí metody Parse() objektu float

Console.WriteLine("\nObvod čtverce o straně {0} je {1}.", strana, 4*strana);
Console.WriteLine("Obsah čtverce o straně {0} je {1}.\n", strana, strana*strana);

Console.Write("Čekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 06 – obsahy obrazců

Úkol

Vytvořte konzolovou aplikaci, která spočítá (podle volby uživatele) obsah čtverce, obdélníku nebo kruhu.

Vypadat to může třeba takto:



Procvičíte si

- načítání číselných hodnot (parsování vstupu)
- podmínky (*if*, *switch*)

Stručný postup

1. Program bude mít tři části – výpočet obsahu čtverce, obdélníka a kruhu. Podle úvodní volby uživatele (zadání celého čísla 1, 2 nebo 3) se provede jedna z nich.

   Můžeme použít ***neúplnou podmínku*** ve tvaru

   ```
   if (podmínka) { větev true }
   ```

   ```
   if (volba == 1) {  // obsah čtverce
     ...
   }
   if (volba == 2) {  // obsah obdélníka
     ...
   }
   if (volba == 3) {  // obsah kruhu
     ...
   }
   ```

Takto ale nepoznáme, že uživatel zadal na začátku nesmyslné číslo volby (například 8). Proto bude lepší použít ***úplný podmíněný příkaz*** ve tvaru

```
if (podmínka) { větev true } else { větev false }
```

```
if (volba == 1) {  // obsah čtverce
  ...
} else {
  if (volba == 2) {  // obsah obdélníka
    ...
  } else {
    if (volba == 3) {  // obsah kruhu
      ...
    } else {
      // nesmyslné číslo volby
    }
  }
}
```

Takto už poznáme, že bylo zadáno chybné číslo volby, a můžeme na to uživatele upozornit. Kód ale díky do sebe vnořeným příkazům ***if*** není příliš přehledný. V tomto případě je výhodné využít příkazu ***switch***, který slouží k ***vícenásobnému větvení***.

```
switch (volba) {
  case 1: // obsah čtverce
    ...
    break;
  case 2: // obsah obdélníka
    ...
    break;
  case 3: // obsah kruhu
    ...
    break;
  default :  // nesmyslné číslo volby
    ...
    break;
}
```

Příkaz ***break*** ukončí provádění dané větve a na další větve už nedojde.
Větev ***default*** se provede v případě, že se neprovedla řádná z větví předcházejících.

V každé větvi může být i více hodnot, například ***case 0, 1, 2: ...***

Hodnota, podle které se příkaz ***switch*** rozhoduje (v našem případě obsah proměnné ***volba***) musí být ordinálního typu, tj. celé číslo nebo znak.

2. Při výpočtu obsahu kruhu můžeme buď zapsat přímo konstantu π jako ***3.14***, nebo pro větší přesnost využít konstanty ze třídy ***Math***, tj. ***Math.PI***.

Řešení

```
Console.WriteLine("Výpočet obsahů obrazců\n");

Console.WriteLine("1 - obsah čtverce");
Console.WriteLine("2 - obsah obdélníka");
Console.WriteLine("3 - obsah kruhu\n");

Console.Write("Zadejte číslo vaší volby: ");
int volba = int.Parse(Console.ReadLine());

Console.WriteLine("\n---------------------------------\n");

switch (volba) {
  case 1: // obsah čtverce
    Console.Write("Zadejte délku strany čtverce: ");
    float strana = float.Parse(Console.ReadLine());
    Console.WriteLine("\nObsah čtverce o straně {0} je {1}.", strana, strana * strana);
    break;

  case 2: // obsah obdélníka
    Console.Write("Zadejte délku první strany obdélníka: ");
    float a = float.Parse(Console.ReadLine());
    Console.Write("Zadejte délku druhé strany obdélníka: ");
    float b = float.Parse(Console.ReadLine());
    Console.WriteLine("\nObsah obdélníka o stranách {0} a {1} je {2}.", a, b, a * b);
    break;

  case 3: // obsah kruhu
    Console.Write("Zadejte poloměr kruhu: ");
    float r = float.Parse(Console.ReadLine());
    Console.WriteLine("\nObsah kruhu o poloměru {0} je {1}.", r, Math.PI * r * r);
    break;

  default :  // nesmyslné číslo volby
    Console.WriteLine("\nZadali jsme nesmyslné číslo volby, musíte zadat 1, 2 nebo 3!");
    break;
}

Console.Write("\nČekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 07 – lineární rovnice

Úkol

Vytvořte konzolovou aplikaci, která vyřeší zadanou lineární rovnici. Nezapomeňte ošetřit případ, kdy lineární rovnice nemá žádné řešení nebo kdy jich má nekonečně mnoho.

Vypadat to může třeba takto:

```
Řešení lineární rovnice ve tvaru Ax + B = 0

Zadejte koeficient A: 2
Zadejte koeficient B: 3

Lineární rovnice 2x + 3 = 0 má právě jedno řešení, x = -1,5.

čekám na stisk libovolné klávesy...
```

Procvičíte si

- načítání číselných hodnot (parsování vstupu)
- podmínky

Stručný postup

Lineární rovnice je rovnice ve tvaru $Ax + B = 0$. Vstupem programu tedy budou dvě čísla A, B – koeficienty této rovnice.

Pokud je $A \neq 0$, rovnice má právě jedno řešení $x = -B / A$.

Pokud je $A = 0$, pak jsou dvě možnosti:

- pokud i $B = 0$, má rovnice nekonečně mnoho řešení (tj. řešením je každé reálné číslo, za x můžeme dosadit cokoliv a rovnice bude vždy splněna),
- pokud $B \neq 0$, nemá rovnice žádné řešení (ať za x dosadíme cokoliv, rovnice nebude nikdy splněna).

Řešení

```
Console.WriteLine("Řešení lineární rovnice ve tvaru Ax + B = 0\n");
Console.Write("Zadejte koeficient A: ");
float A = float.Parse(Console.ReadLine());
Console.Write("Zadejte koeficient B: ");
float B = float.Parse(Console.ReadLine());

Console.WriteLine();

if (A != 0) {
  // rovnice má právě jedno řešení
  Console.WriteLine("Lineární rovnice {0}x + {1} = 0 má právě jedno řešení,
x = {2}.", A, B, -B/A);
} else {
  if (B == 0) {
    // nekonečně mnoho řešení, A==0, B==0
    Console.WriteLine("Lineární rovnice {0}x + {1} = 0 má nekonečně mnoho
řešení.", A, B);
  } else {
    // žádné řešení, A==0, B!=0
    Console.WriteLine("Lineární rovnice {0}x + {1} = 0 nemá žádné řešení.", A, B);
  }
}

Console.Write("\nČekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 08 – kvadratická rovnice

Úkol

Vytvořte konzolovou aplikaci, která vyřeší zadanou kvadratickou rovnici. Nezapomeňte ošetřit případ, kdy kvadratická rovnice nemá žádné řešení. Pozor také na to, že pokud je koeficient u x2 nulový, nejedná se o kvadratickou rovnici.

Vypadat to může třeba takto:

```
Řešení kvadratické rovnice ve tvaru Ax^2 + Bx + C = 0

Zadejte koeficient A: 2
Zadejte koeficient B: 7
Zadejte koeficient C: -15

Rovnice 2x^2 + 7x + -15 = 0 má dva různé reálné kořeny, x1 = -5, x2 = 1,5.

čekám na stisk libovolné klávesy...
```

Poznámka: Při výpisu záporného koeficientu následují za sebou znaménka + a –, takto by to zapsané být nemělo, ale tím se nyní nezabývejte.

Procvičíte si

- matematické výpočty
- podmínky

Stručný postup

Kvadratická rovnice je rovnice ve tvaru $Ax^2 + Bx + C = 0$. Vstupem programu tedy budou tři čísla ***A***, ***B***, ***C*** – koeficienty této rovnice.

Pokud je $A = 0$, nejedná se o kvadratickou rovnici a nebudeme se jí zde zabývat (případně můžete převzít řešení z předchozího příkladu, ale potom pozor na názvy proměnných!).

Pokud je $A \neq 0$, jedná se o kvadratickou rovnici a nejprve je potřeba vypočítat diskriminant

$$D = B^2 - 4AC$$

V jazyce C# musíme výpočet zapsat takto

```
D = Math.Pow(B,2) - 4*A*C
```

Metoda ***Pow*** třídy ***Math*** umocní první parametr na exponent zadaný jako druhý parametr. Všechny proměnné deklarujeme typu double, protože metody třídy ***Math*** většinou vrací datový typ ***double***, tak abychom se nemuseli starat o konverzi typu ***float*** na ***double*** a naopak.

Při umocňování na druhou bychom mohli místo ***Math.Pow(B,2)*** použít i ***B*B***.

Hodnota diskriminantu určuje počet řešení kvadratické rovnice,

- pokud $D > 0$, má rovnice dva různé reálné kořeny, které spočítáme $x_{1,2} = \frac{-B \pm \sqrt{D}}{2A}$

  v jazyce C# musíme výpočet zapsat takto

  `(-B-Math.Sqrt(D))/(2*A)` a `(-B+Math.Sqrt(D))/(2*A)`

- pokud $D = 0$, má rovnice jeden dvojnásobný kořen, který spočítáme $x = \frac{-B}{2A}$

  v jazyce C# musíme výpočet zapsat takto

  `(-B)/(2*A)`

- pokud $D < 0$, nemá rovnice reálné řešení.

Řešení

```
Console.WriteLine("Řešení kvadratické rovnice ve tvaru Ax^2 + Bx + C = 0\n");
Console.Write("Zadejte koeficient A: ");
double A = double.Parse(Console.ReadLine());
Console.Write("Zadejte koeficient B: ");
double B = double.Parse(Console.ReadLine());
Console.Write("Zadejte koeficient C: ");
double C = double.Parse(Console.ReadLine());
Console.WriteLine();

if (A == 0) {
  // nejedná se o kvadratickou rovnici
  Console.WriteLine("Rovnice {0}x^2 + {1}x + {2} = 0 není rovnicí kvadratickou,
tu nebudu řešit!", A, B, C);
} else {
  // zde již víme, že A!=0, jedná se tedy o kvadratickou rovnici
  double D = Math.Pow(B, 2) - 4*A*C;  // vypočítáme diskriminant
  if (D > 0) {    // rovnice má dva různé reálné kořeny
    Console.WriteLine("Rovnice {0}x^2 + {1}x + {2} = 0 má dva různé reálné
kořeny, x1 = {3}, x2 = {4}.", A, B, C, (-B-Math.Sqrt(D))/(2*A), (-
B+Math.Sqrt(D))/(2*A));
  }
  if (D == 0) {    // rovnice má jeden dvojnásobný kořen
    Console.WriteLine("Rovnice {0}x^2 + {1}x + {2} = 0 má jeden dvojnásobný
kořen, x = {3}.", A, B, C, (-B)/(2*A));
  }
  if (D < 0) {    // rovnice nemá žádné reálné řešení
    Console.WriteLine("Rovnice {0}x^2 + {1}x + {2} = 0 nemá reálné kořeny.", A, B, C);
  }
}
Console.Write("\nČekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 09 – trojúhelník

Úkol

Vytvořte konzolovou aplikaci, která načte délky tří úseček a vypíše, zda z těchto úseček lze sestrojit trojúhelník.

Nápověda: Vzpomeňte si na trojúhelníkovou nerovnost, případně se podívejte na *http://it.pedf.cuni.cz/~proch/program/sss.htm*.

Vypadat to může třeba takto:

```
Test, zda lze sestrojit trojúhelník

Zadejte délku úsečky a: 1
Zadejte délku úsečky b: 5
Zadejte délku úsečky c: 6

Z úseček délek 1, 5 a 6 NELZE sestrojit trojúhelník.

čekám na stisk libovolné klávesy...
```

Procvičíte si

- složené podmínky

Stručný postup

1. Aby bylo možno z úseček délek a, b, c sestrojit trojúhelník, musí platit:
   Součet délek *libovolných* dvou úseček je větší než délka úsečky třetí.

   Důležité je slovíčko „libovolných“ – aby bylo možno trojúhelník sestrojit, nestačí jen aby platilo $a + b > c$, ale musí také platit $a + c > b$ i $b + c > a$.

2. S jednoduchými do sebe vnořenými podmínkami úlohu vyřešíme, není to ale příliš přehledné.

```
if (a + b > c) {
  if (a + c > b) {
    if (b + c > a) {
      // lze sestrojit, všechny tři nerovnosti platí
    } else {
      // nelze sestrojit, jedna z nerovností neplatí
    }
  } else {
    // nelze sestrojit, jedna z nerovností neplatí
  }
} else {
  // nelze sestrojit, jedna z nerovností neplatí
}
```

3. Když použijete ***složené podmínky***, kód se výrazně zjednoduší a zpřehlední.

   Logické spojky se v C# zapisují takto:

   ```
   ! – negace,
   && – a současně (AND),
   || – nebo (OR).
   ```

   Logické spojky mají menší prioritu než relační operátory, proto lze napsat

   ```
   if (A < B && B < C) {...}
   ```

   tj. není potřeba jednotlivé podmínky dávat do závorek, pro někoho to ale může být přehlednější

   ```
   if ( (A < B) && (B < C) ) {...}
   ```

   Pozor, některé programovací jazyky (například Pascal) to mají přesně naopak – logické spojky mají větší prioritu než relační operátory, tam je uzávorkování jednotlivých podmínek nutné!

   Při použití složených podmínek si vystačíme s jediným příkazem **`if`**.

```
if (a + b > c && a + c > b && b + c > a) {
  // lze sestrojit, všechny tři nerovnosti platí
} else {
  // nelze sestrojit, jedna z nerovností neplatí
}
```

Řešení

```
Console.WriteLine("Test, zda lze sestrojit trojúhelník\n");

Console.Write("Zadejte délku úsečky a: ");
float a = float.Parse(Console.ReadLine());

Console.Write("Zadejte délku úsečky b: ");
float b = float.Parse(Console.ReadLine());

Console.Write("Zadejte délku úsečky c: ");
float c = float.Parse(Console.ReadLine());

Console.WriteLine();

if (a + b > c && a + c > b && b + c > a) {
  // lze sestrojit, všechny tři nerovnosti platí
  Console.WriteLine("Z úseček délek {0}, {1} a {2} LZE sestrojit trojúhelník.", a, b, c);
} else {
  // nelze sestrojit, jedna z nerovností neplatí
  Console.WriteLine("Z úseček délek {0}, {1} a {2} NELZE sestrojit trojúhelník.", a, b, c);
}
Console.Write("\nČekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 10 – výpočet BMI

Úkol

Vytvořte konzolovou aplikaci, která na základě zadané výšky (v metrech) a váhy (v kilogramech) osoby vypočte její index BMI a vypíše, do které kategorie podváhy/nadváhy podle tohoto indexu osoba patří.

Nápověda: BMI = Body Mass Index, index tělesné hmotnosti se vypočte

$$BMI = \frac{váha\ v\ kg}{(výška\ v\ m)^2}$$

Tabulku se stupni podváhy/nadváhy najdete například na ***https://cs.wikipedia.org/wiki/Index_t%C4%9Blesn%C3%A9_hmotnosti***.

Kategorie	Rozsah BMI
těžká podvýživa	⩽ 16,5
podváha	16,5 – 18,5
ideální váha	18,5 – 25
nadváha	25 – 30
mírná obezita	30 – 35
střední obezita	35 – 40
morbidní obezita	> 40

V této tabulce jsou uváděna rozmezí pro jednotlivé stupně podváhy/nadváhy, pokud je tedy $25 < BMI \leq 30$, jedná se o nadváhu…

Vypadat to může třeba takto:

```
Výpočet Body Mass Indexu (BMI)

Zadejte vaši výšku (v m): 1,79
Zadejte vaši váhu (v kg): 90

Při výšce 1,79 m a váze 90 kg je váš BMI roven 28,09.
Máte nadváhu.

čekám na stisk libovolné klávesy...
```

Procvičíte si

- vícenásobné podmínky
- formátování výstupu – formátovací řetězce

Stručný postup

1. Vlastní výpočet indexu BMI je jednoduchý, komplikovanější je to s následným vyhodnocením. Můžeme vnořovat příkazy ***if*** do sebe tak, jak jsme zvyklí – pokud zdrojový kód zformátujeme následujícím způsobem, bude to i přehledné. Uvědomte si, že příkaz ***switch*** nám zde nepomůže.

```
if (BMI <= 16.5) {
  Console.WriteLine("POZOR, máte těžkou podvýživu!");
}
else if (BMI <= 18.5) {
  Console.WriteLine("Máte podváhu.");
}
else if (BMI <= 25) {
  Console.WriteLine("Gratulujeme, máte ideální váhu!");
}
else if (BMI <= 30) {
  Console.WriteLine("Máte nadváhu.");
}
```

...a tak dále

2. Pokud pro výpis použijeme příkaz

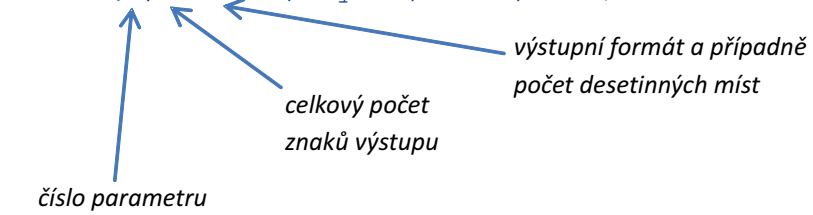
```
Console.WriteLine("Při výšce {0} m a váze {1} kg je váš BMI roven
{2}.", vyska, vaha, BMI);
```

bude BMI index zbytečně vypsán na plný počet desetinných míst:

Při výšce 1,79 m a váze 90 kg je váš BMI roven 28,0890109547143.

K zástupným symbolům můžeme připsat informaci,
v jakém formátu se má daný parametr vypsat, tzv. ***formátovací řetězec***.
Například pro znak měny ***C***, pro desetinný formát ***D***, pro vědecký formát ***E***,
pro formát s pevnou desetinnou čárkou ***F***, pro číselný formát ***N***,
pro procentuální formát ***P*** a další,
viz ***https://msdn.microsoft.com/en-us/library/dwhawy9k%28v=vs.110%29.aspx***.

```
Console.WriteLine("Při výšce {0} m a váze {1} kg je váš BMI
roven {2,8:F2}.", vyska, vaha, BMI);
```



tedy vypíše:

Při výšce 1,79 m a váze 90 kg je váš BMI roven 28,08.

Řešení

```
Console.WriteLine("Výpočet Body Mass Indexu (BMI)\n");

Console.Write("Zadejte vaši výšku (v m): ");
double vyska = double.Parse(Console.ReadLine());

Console.Write("Zadejte vaši váhu (v kg): ");
double vaha = double.Parse(Console.ReadLine());

double BMI = vaha / (vyska * vyska);

Console.WriteLine("\nPři výšce {0} m a váze {1} kg je váš BMI roven {2,4:F2}.",
vyska, vaha, BMI);

if (BMI <= 16.5) {
  Console.WriteLine("POZOR, máte těžkou podvýživu!");
}
else if (BMI <= 18.5) {
  Console.WriteLine("Máte podváhu.");
}
else if (BMI <= 25) {
  Console.WriteLine("Gratulujeme, máte ideální váhu!");
}
else if (BMI <= 30) {
  Console.WriteLine("Máte nadváhu.");
}
else if (BMI <= 35) {
  Console.WriteLine("Máte mírnou obezitu!");
}
else if (BMI <= 40) {
  Console.WriteLine("Máte střední obezitu!!");
}
else {  // zde už musí být BMI > 40
  Console.WriteLine("Máte morbidní obezitu!!!");
}

Console.WriteLine("\nČekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 11 – součet a průměr

Úkol

Vytvořte konzolovou aplikaci, která bude načítat celá čísla a poté spočítá a vypíše jejich počet, součet a průměr. Načítání bude ukončeno zadáním čísla 999, to se už mezi zadaná čísla počítat nebude.

Vypadat to může třeba takto:

```
Součet a průměr zadaných čísel
Zadávejte celá čísla, zadávání ukončíte číslem 999

Zadejte celé číslo: 2
Zadejte celé číslo: 5
Zadejte celé číslo: 10
Zadejte celé číslo: 999

Počet zadaných čísel: 3, součet: 17, průměr: 5,666667

Čekám na stisk libovolné klávesy...
```

Procvičíte si

- cyklus s podmínkou na začátku
- operátory ++, --, +=, …
- přetypování (konverze číselných hodnot)

Stručný postup

1. Načítání čísel pomocí cyklu s podmínkou na začátku je jednoduché.

```
int soucet = 0, pocet = 0;  // inicializace
int cislo = int.Parse(Console.ReadLine());  // načtu první číslo

while (cislo != 999) {  // dokud není zadáno číslo 999, opakuj
  pocet++;              // zvětši počet čísel
  soucet += cislo;      // a přičti zadané číslo k součtu

  cislo = int.Parse(Console.ReadLine());  // načti další číslo
}
```

Hodnotu proměnné ***pocet*** můžete zvětšovat o 1 buď klasickým přiřazovacím příkazem ve tvaru `pocet = pocet + 1`

nebo můžeme využít zkráceného zápisu `pocet += 1`

nebo jen `pocet++`

Podobně přičítání dalšího čísla do proměnné ***soucet***
můžeme místo `soucet = soucet + cislo`

zkráceně zapsat `soucet += cislo`.

Podobně lze zapsat `pocet--` `soucin *= cislo` `++pocet`

Rozdíl mezi `pocet++` a `++pocet` je v tom, že zápis `pocet++` nejprve vrátí hodnotu proměnné ***pocet*** a poté proměnnou ***pocet*** zvětší o ***1***.

Naopak `++pocet` nejprve zvětší hodnotu proměnné ***pocet*** o ***1*** a poté tuto hodnotu vrátí.

Tedy

```
int pocet = 1;
int A = 0;

A = pocet++;
```

V proměnné `A` bude hodnota `1`.
V proměnné `pocet` bude hodnota `2`.

```
int pocet = 1;
int B = 0;

B = ++pocet;
```

V proměnné `B` bude hodnota `2`.
V proměnné `pocet` bude hodnota `2`.

2. Pozor na následující „záludnost“ při výpisu vypočítaného průměru.

   Pokud použijeme příkaz ve tvaru

   ```
   Console.WriteLine("průměr: {0}", soucet/pocet);
   ```

   a pokud načteme čísla ***1*** a ***2***, bude výsledný průměr ***(1+2)/2 = 1,5***,
   ale vypíše se pouze ***1***. Důvodem je, že proměnné ***soucet*** a ***pocet*** jsou celočíselné,
   proto C# usoudí, že operátor mezi nimi je celočíselné dělení,
   proto bude vypsán výsledek ***1*** a ne ***1,5***.

   Musíme pomocí přetypování sdělit, že chceme,
   aby byl výraz vypočítán v reálných číslech

   ```
   Console.WriteLine("průměr: {0}", (float)soucet/pocet);
   ```

   Přetypování má obecně tvar `(nový typ)proměnná`, například

   ```
   int celeCislo = 3;
   float realneCislo = 2.57F;
   ```

Přiřazení `realneCislo = celeCislo;` lze provést, zde se provede implicitní konverze datového typu ***int*** na ***float***.

U přiřazení `celeCislo = realneCislo;` však překladač zahlásí chybu, že nelze automaticky převést typ ***float*** na ***int***. Musíme použít přetypování ve tvaru

```
celeCislo = (int)realneCislo;
```

Abychom byli úplně přesní, ve výraze `(float)soucet/pocet` dojde k přetypování na typ ***float*** pouze u proměnné ***soucet*** (přetypování má větší prioritu než dělení), ale to už stačí k tomu, aby překladač dělení provedl jako reálné a ne celočíselné.

Řešení

```
Console.WriteLine("Součet a průměr zadaných čísel");
Console.WriteLine("Zadávejte celá čísla, zadávání ukončíte číslem 999\n");

int soucet = 0, pocet = 0;  // inicializace, zatím žádná čísla nebyla zadána

Console.Write("Zadejte celé číslo: ");
int cislo = int.Parse(Console.ReadLine());  // načtu první číslo

while (cislo != 999) {  // dokud není zadáno číslo 999, opakuj
  pocet++;              // zvětši počet čísel
  soucet += cislo;      // a přičti zadané číslo k součtu

  Console.Write("Zadejte celé číslo: ");
  cislo = int.Parse(Console.ReadLine());    // načti další číslo
}

if (pocet > 0) {  // podmínka je nutná proto, abychom nedělili nulou!
  Console.WriteLine("\nPočet zadaných čísel: {0}, součet: {1}, průměr: {2}",
pocet, soucet, (float)soucet/pocet);
} else {
  Console.WriteLine("\nNebylo zadáno žádné číslo!");
}

Console.WriteLine("\nčekám na stisk libovolné klávesy...");
Console.ReadKey();
```

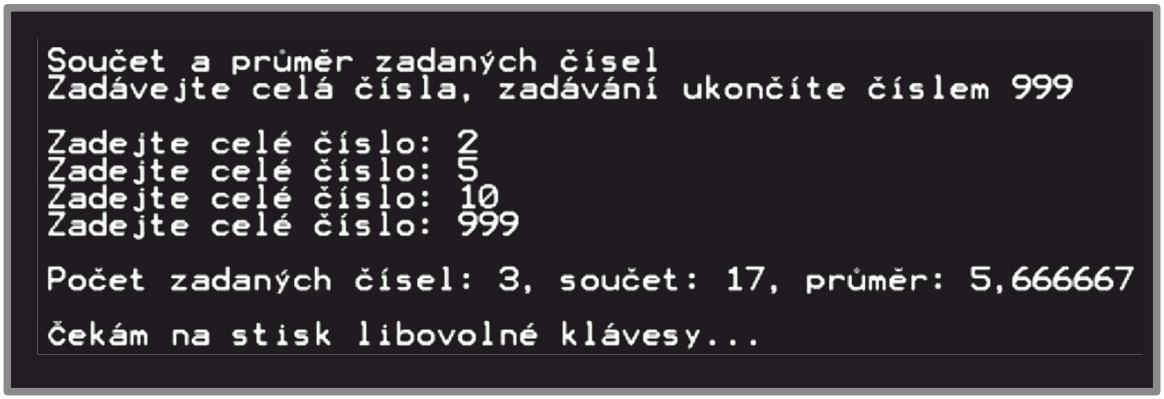
Cvičení 12 – součet a průměr

Úkol

Vytvořte konzolovou aplikaci, která bude načítat celá čísla a poté spočítá a vypíše jejich počet, součet a průměr. Načítání bude ukončeno zadáním čísla 999, to se už mezi zadaná čísla počítat nebude.

Využijte cyklus s podmínkou na konci, který je vhodnější právě pro takovouto situaci, tj. při načítání hodnot od uživatele. Jinak zůstává zadání stejné jako v předcházející úloze.

Vypadat to může třeba takto:



Procvičíte si

- cyklus s podmínkou na konci

Stručný postup

1. Při načítání čísel od uživatele není využití cyklu s podmínkou na začátku příliš vhodné, povšimněte si, že jeden příkaz se tam opakuje 2.

```
int soucet = 0, pocet = 0;  // inicializace
int cislo = int.Parse(Console.ReadLine());  // načtu první číslo

while (cislo != 999) {  // dokud není zadáno číslo 999, opakuj
  pocet++;              // zvětši počet čísel
  soucet += cislo;      // a přičti zadané číslo k součtu

  cislo = int.Parse(Console.ReadLine());  // načti další číslo
}
```

2. V této situaci je výhodnější použít cyklus s podmínkou na konci ve tvaru
   `do {tělo cyklu} while (podmínka)`

   Tento cyklus se provádí, dokud podmínka platí (tj. dokud je splněna).
   Nejprve se provede tělo cyklu, pak se vyhodnotí podmínka.
   Pokud je podmínka splněna, opět se provede tělo cyklu…

```
do {
  Console.Write("Zadejte celé číslo: ");
  cislo = int.Parse(Console.ReadLine());    // načti další číslo

  pocet++;              // zvětši počet čísel
  soucet += cislo;      // a přičti zadané číslo k součtu
} while (cislo != 999); // dokud není zadáno číslo 999, opakuj
```

Použitím cyklu s podmínkou na konci jsme ušetřili jeden příkaz (načítání čísla před tělem cyklu), ale číslo ***999***, kterým se načítání ukončí, se nám vždy započítá mezi načtená čísla – to je pochopitelné, protože podmínka pro ukončení cyklu se vyhodnotí až po načtení a zpracování čísla.
Toto je potřeba v programu ošetřit, nezapomeňte na to!

```
if (pocet > 1) {
  pocet--;
  soucet -= 999;
  Console.WriteLine("\nPočet zadaných čísel: {0}, součet: {1},
průměr: {2}", pocet, soucet, (float)soucet/pocet);
} else {
  Console.WriteLine("\nNebylo zadáno žádné číslo!");
}
```

Také bychom mohli postupovat jinak, a to tak, že bychom na začátku proměnné soucet a pocet inicializovali následovně

```
int soucet = -999, pocet = -1;
```

Následující cyklus by zůstal stejný a závěrečná podmínka by byla jednodušší

```
if (pocet > 1)
  Console.WriteLine("\nPočet zadaných čísel: {0}, součet: {1},
průměr: {2}", pocet, soucet, (float)soucet/pocet);
} else {
  Console.WriteLine("\nNebylo zadáno žádné číslo!");
}
```

Řešení

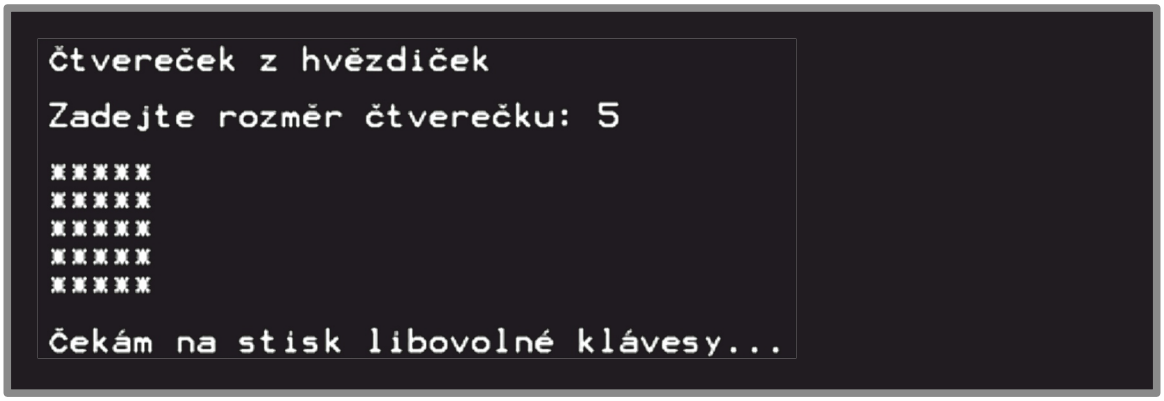
```
Console.WriteLine("Součet a průměr zadaných čísel");
Console.WriteLine("Zadávejte celá čísla, zadávání ukončíte číslem 999\n");

int soucet = 0, pocet = 0;  // inicializace, zatím žádná čísla nebyla zadána
int cislo; // deklarace proměnné

do {
  Console.Write("Zadejte celé číslo: ");
  cislo = int.Parse(Console.ReadLine());    // načti další číslo

  pocet++;              // zvětši počet čísel
  soucet += cislo;      // a přičti zadané číslo k součtu
} while (cislo != 999); // dokud není zadáno číslo 999, opakuj

if (pocet > 1) {  // podmínka je nutná proto, abychom nedělili nulou!
  pocet--;        // musíme upravit, číslo 999 se nám započítá do počtu
  soucet -= 999;  // i součtu
  Console.WriteLine("\nPočet zadaných čísel: {0}, součet: {1}, průměr: {2}",
pocet, soucet, (float)soucet/pocet);
} else {
  Console.WriteLine("\nNebylo zadáno žádné číslo!");
}

Console.WriteLine("\nČekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 13 – hvězdičky

Úkol

Vytvořte konzolovou aplikaci, která pro zadané kladné celé číslo *N* (vstup od uživatele) vykreslí na konzoli čtvereček o rozměrech *NN* vytvořený z hvězdiček. Využijte cyklu ***for***.

Vypadat to může třeba takto:



Procvičíte si

- cyklus for

Stručný postup

1. Realizace pomocí dvou do sebe vnořených ***while*** cyklů by mohla vypadat například takto

```
int y = 1;          // řídící proměnná prvního cyklu
while (y <= N) {    // cyklus pro nakreslení N řádků
  int x = 1;        // řídící proměnná druhého cyklu
  while (x <= N) { // cyklus pro nakreslení N hvězdiček v jednom
řádku
    Console.Write("*");  // kreslím hvězdičky do řádku
    x++;
  }
  y++;
  Console.WriteLine();   // přechod na další řádek
}
```

Všimněte si, že každý z cyklů má svoji vlastní řídící proměnnou (cyklus pro kreslení řádků proměnnou ***y***, cyklus pro kreslení hvězdiček na jednom řádku proměnnou ***x***).

2. Stručnější a přehlednější řešení bude s využitím ***for*** cyklu

```
for (inicializace; podmínka; příkaz) {tělo cyklu}
```

Nejprve se, ještě před začátkem cyklu, provede ***inicializace***,
zde lze například i deklarovat potřebnou řídící proměnnou cyklu.

Tělo cyklu se provádí tak dlouho, dokud je ***podmínka*** splněna.

Po každém provedení těla cyklu se ještě provede ***příkaz***,
zde se často zvětšuje/zmenšuje hodnota řídící proměnné cyklu.

Vidíme, že ***for*** cyklus je vlastně zkratkou za následující ***while*** cyklus

```
inicializace;
while (podmínka) {
  tělo cyklu
  příkaz;
}
```

Řešení

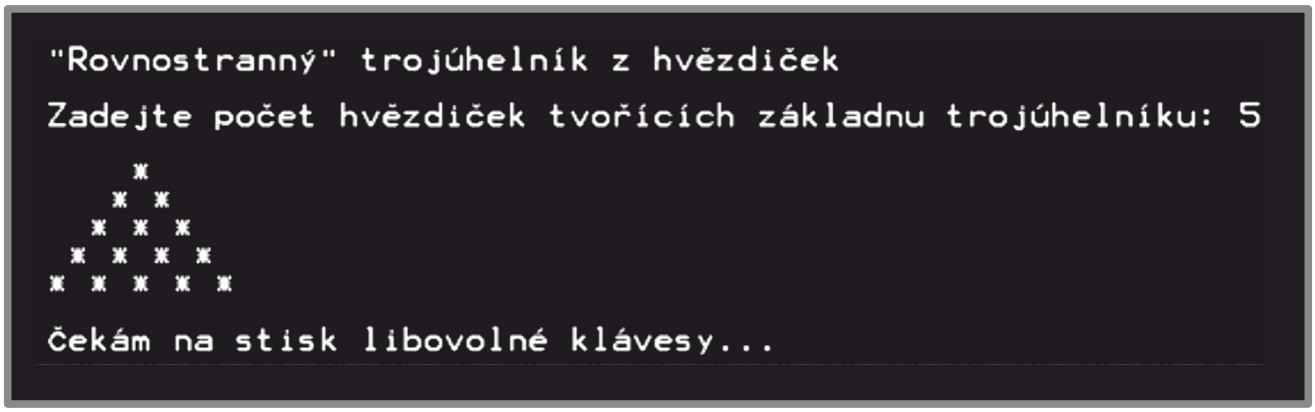
```
Console.WriteLine("Čtvereček z hvězdiček\n");

Console.Write("Zadejte rozměr čtverečku: ");
int N = int.Parse(Console.ReadLine());

Console.WriteLine();

for (int y = 1; y <= N; y++) {  // cyklus pro nakreslení N řádků,
                                // řídící proměnná cyklu je y
  for (int x = 1; x <= N; x++) {// cyklus pro nakreslení N hvězdiček
                                // v jednom řádku, řídící proměnná cyklu je x
    Console.Write("*");  // kreslím hvězdičky do řádku
  }
  Console.WriteLine();   // přechod na další řádek
}

Console.WriteLine("\nČekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 14 – trojúhelník z hvězdiček

Úkol

Vytvořte konzolovou aplikaci, která pro zadané kladné celé číslo N (vstup od uživatele) vykreslí na konzoli z hvězdiček „rovnostranný trojúhelník“ o základně tvořené N hvězdičkami. Využijte cyklu ***for***.

Vypadat to může třeba takto:



Procvičíte si

- cyklus ***for***, vnořování cyklů do sebe

Stručný postup

1. Základem budou opět dva do sebe vnořené ***for*** cykly

```
for (int y = 1; y <= N; y++) {
  for (int x = 1; x <= y; x++) {  // na každém řádku nakreslíme
                  // o jednu hvězdičku více než na řádku předchozím
    Console.Write("* ");  // mezi každými dvěma hvězdičkami musí
                          // být vynechaná mezera
  }
Console.WriteLine();
}
```

Takto ale nakreslíme „pravoúhlý trojúhelník“

2. Proto musíme na začátku každého řádku ještě vynechat potřebný počet mezer, k tomu využijeme ještě jeden ***for*** cyklus. Pozor na pojmenovávání řídících proměnných cyklů, abyste se do toho nezamotali.

Řešení

```
Console.WriteLine("\"Rovnostranný\" trojúhelník z hvězdiček\n");

Console.Write("Zadejte počet hvězdiček tvořících základnu trojúhelníku: ");
int N = int.Parse(Console.ReadLine());

Console.WriteLine();

for (int y = 1; y <= N; y++) {
  for (int m = 1; m <= N-y; m++) {
    Console.Write(" ");    // na začátku každého řádku vynecháme
                           // potřebný počet mezer
  }
  for (int x = 1; x <= y; x++) {  // na každém řádku nakreslíme o jednu
                                  // hvězdičku více než na řádku předchozím
    Console.Write("* ");   // mezi každými dvěma hvězdičkami
                           // musí být vynechaná mezera
  }
  Console.WriteLine();
}

Console.WriteLine("\nČekám na stisk libovolné klávesy...");
Console.ReadKey();
```

Cvičení 15 – faktoriál

Úkol

Vytvořte konzolovou aplikaci, která pro zadané přirozené číslo ***N*** vypočítá ***N!*** (tj. ***N faktoriál***). Úlohu vyřešte pomocí

a) ***for*** cyklu,

b) cyklu s podmínkou na začátku (***while*** ...),

c) cyklu s podmínkou na konci (***do*** ... ***while***).

Zvolte vhodný datový typ tak, aby bylo možno vypočítat faktoriál co největšího čísla. Porovnejte z tohoto hlediska různé datové typy.

Vypadat to může třeba takto:

```
Výpočet faktoriálu
Zadejte N: 27
27! = 10888869450418352160768000000
Čekám na stisk libovolné klávesy...
```

Nápověda: Faktoriál přirozeného čísla ***N*** je součin všech přirozených čísel od ***1*** až do ***N***.

Tedy $N! = 1 \cdot 2 \cdot \cdots \cdot (N-1) \cdot N$

Tedy 1! = 1
2! = 12
3! = 123
4! = 1234
5! = 12345

Procvičíte si

- různé druhy cyklů a jejich porovnání
- porovnání rozsahu různých číselných datových typů

Stručný postup

1. ***For*** cyklus

   Faktoriál můžeme počítat přesně podle definice, budeme k tomu potřebovat jednu pomocnou proměnnou ***i***, která bude nabývat hodnot ***1, 2, ..., N***.

```
faktorial = 1;

for(uint i = 1; i <= N; i++) {
  faktorial = faktorial * i;
}
```

   Zápis těla cyklu můžeme ještě zkrátit na `faktorial *= i;`

   Nebo můžeme pomocnou proměnnou ***i*** ušetřit a měnit přímo hodnoty proměnné ***N***, tedy počítat faktoriál „odzadu“, ***N! = N(N–1) … 21***

```
for(; N > 0; N--) {
  faktorial *= N;
}
```

   Toto je spíše ukázka toho, jak „stručně“ se dá v C# také psát, není to ale příliš přehledné a hlavně měníme hodnotu proměnné ***N*** – po ukončení cyklu bude vždy ***N == 0***.
   Nebudeme moci tedy vypsat výsledek výpočtu pomocí

```
Console.WriteLine("\n{0}! = {1}", N, faktorial);
```

2. Cyklus s podmínkou na začátku (***while ...***)

```
uint i = 1;
while (i <= N) {
  faktorial = faktorial * i;
  i++;
}
```

3. Cyklus s podmínkou na konci (***do ... while***)

```
uint i = 1;
do {
  faktorial = faktorial * i;
  i++;
} while (i <= N);
```

4. Pokud jako vstup zadáme ***0***, výsledek bude ***1***. To je v pořádku, v matematice se definuje ***0! = 1***.

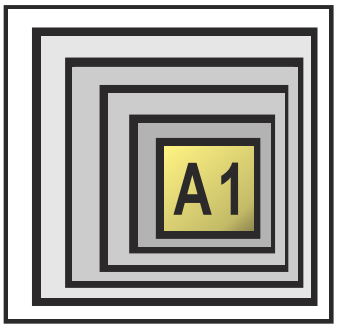
5. Při experimentování s deklarací proměnné ***faktorial*** zjistíte, že pokud bude typu
   - ***uint***, program spočítá nejvýše ***12!***
   - ***ulong***, program spočítá nejvýše ***20!***
   - ***decimal***, program spočítá nejvýše ***27!***
   - ***double***, program sice spočítá i ***170!***, ale ne přesně, jen na ***15*** platných cifer

Řešení

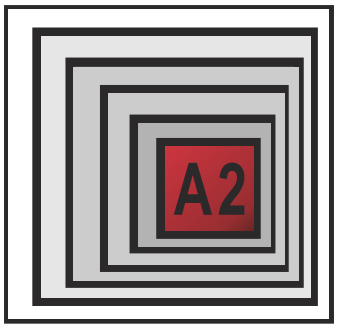
V řešení jsou použity všechny tři druhy cyklů, dva jsou zakomentovány.

```
Console.WriteLine("Výpočet faktoriálu\n");

Console.Write("Zadejte N: ");
uint N = uint.Parse(Console.ReadLine());
double faktorial = 1;

/*
// faktoriál pomocí for cyklu
for(uint i = 1; i <= N; i++) {
  faktorial = faktorial * i;
}
*/

/*
// faktoriál pomocí cyklu s podmínkou na začátku
uint i = 1;
while (i <= N) {
  faktorial = faktorial * i;
  i++;
}
*/

// faktoriál pomocí cyklu s podmínkou na konci
uint i = 1;
do {
  faktorial = faktorial * i;
  i++;
} while (i <= N);

Console.WriteLine("\n{0}! = {1}", N, faktorial);

Console.WriteLine("\nČekám na stisk libovolné klávesy...");
Console.ReadKey();
```

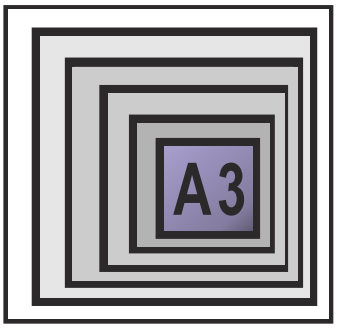
Kantor Ideál



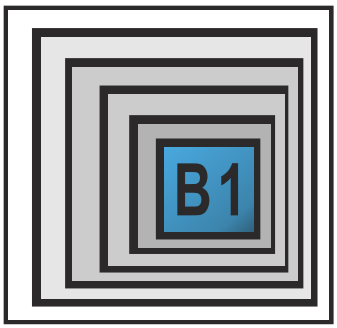
VZDĚLÁVÁNÍ ŘEDITELŮ



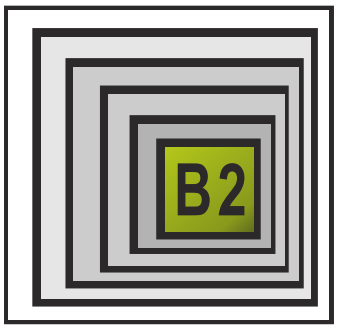
MENTORING



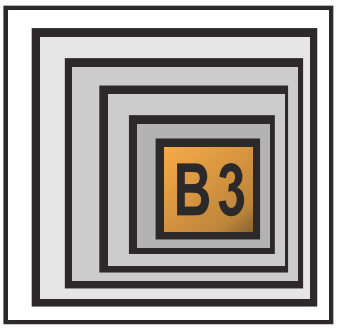
METODIK ICT VE ŠKOLE



CO UŽ MÁME



CO CHCEME



OBOROVÉ DIDAKTIKY

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ