Algoritmy

71 Zjištění, zda je číslo sudé

Chceme-li zjistit, zda je určité číslo sudé, či liché, vypočítáme pomocí binárního operátoru „%“ jeho zbytek po dělení číslem 2. Je-li roven 0, je číslo sudé, je-li roven 1 , je liché.

```
bool JeSude(int cislo)
{
  return cislo % 2 == 0;
}
```

72 Zjištění, zda je číslo dělitelné jiným beze zbytku

Jestliže chceme zjistit, zdali je číslo beze zbytku dělitelné jiným, čili zda je jejich zbytek po dělení roven 0, použijeme binární operátor „%“.

```
bool JeDelitelneZcela(int delenec, int delitel)
{
  if (delenec % delitel == 0)
    return true;
  return false;
}
```

73 Jak získat pole všech prvočísel v intervalu od 2 do N

Eratosthenovo síto je algoritmus určený pro nalezení všech prvočísel v intervalu 2–N. Následující metoda představuje jeho implementaci v jazyce C#. Množina nalezených prvočísel je vrácena jako instance kolekce System.Collections.ArrayList.

```
ArrayList EratosthenovoSito(int MaxN)
{
    ArrayList arl = new ArrayList();
    bool[] pole = new bool[MaxN + 1];
    int i, j;
    for (i = 2; i < MaxN; i++)
        pole[i] = true;
    i = 2;
    while (i * i <= MaxN)
    {
       if (pole[i])
       {
          j = 2;
          while (i * j <= MaxN)
          {
            pole[j * i] = false; j++;
          }
      }
      i++;
   }
```

```
    int m = 0;
    foreach (bool b in pole)
    {
      if (b)
       arl.Add(m);  m++;
    }
    return arl;
}
```

74 Zjištění minimální a maximální hodnoty číselného typu

Číselné datové typy jako int, double, decimal, char apod. obsahují konstantní položky MinValue a MaxValue, které vrací hodnotu představující minimum a maximum intervalu čísel, ve kterém je daný typ platný a který bychom neměli přesáhnout. Pokud se tak stane, bude vyvolána výjimka System.OutOfRangeException.

```
Console.WriteLine("Nejmenší hodnota typu int je: {0}", int.MinValue);
Console.WriteLine("Největší hodnota typu int je: {0}", int.MaxValue);
```

Vypíše:

```
Nejmenší hodnota typu int je -2147483648
Největší hodnota typu int je 2147483647
```

75 Vytvoření instance číselného datového typu z řetězce

Všechny číselné datové typy, jako například int, char, double apod., obsahují statickou metodu Parse, která z řetězce, jenž je jejím argumentem, vytvoří proměnnou daného datového typu. Pokud řetězec neobsahuje správnou reprezentaci čísla, metoda Parse vyvolá výjimku System.FormatException.

```
string s = "12345"
int c = int.Parse(s);
```

76 Jak z řetězce získat číslo zapsané s oddělovačem tisíců

Máme-li řetězec obsahující číslo zapsané s oddělovači tisíců (tedy podle lokálních zvyklostí) převést na číselný datový typ (například int), použijeme přetíženou metodu Parse daného datového typu. Jako její první argument dosadíme daný řetězec a jako druhý pak položku AllowThousands výčtového typu System.Globalization.NumberStyles, která metodě říká, že jsou jednotlivá trojčíslí v řetězci oddělena odpovídajícím znakem.

```
string cislo = "100 000 000";
int m = int.Parse(cislo,
System.Globalization.NumberStyles.AllowThousands);
```

77 Jak zaokrouhlit číslo nahoru

Metoda Ceiling statické třídy System.Math zaokrouhlí číslo na nejbližší vyšší celé číslo. Pokud tedz jako argument zadáme číslo x, které není celé, bude nám vráceno nejbližší následující celé číslo.

```
double cislo = 1.000001;
double zaokNahoru = Math.Ceiling(cislo);  // Vrátí 2
```

78 Jak vypočítat sinus úhlu vyjádřeného ve stupních

Následující kód vypočítá hodnotu sinu určitého úhlu, jehož velikost je zadána ve stupních. Ten je v následujícím případě představován proměnnou stupneUhlu.

```
int stupneUhlu = 45;
// Metoda Math.Sin vypočítá hodnotu sinu úhlu zadaného v radiánech,
// pokud je zadán ve stupních, musíme ho nejprve na radiány převést:
double uhelVRadianech = (Math.PI * stupneUhlu) / 180;
double sinUhlu = Math.Sin(uhelVRadianech);
Console.WriteLine("Sinus úhlu {0} je {1}", stupneUhlu, sinUhlu);
```

V tomto případě bude vypsáno:

```
Sinus úhlu 45 je 0,707106781186547.
```

79 Jak vypočítat kosinus úhlu vyjádřeného ve stupních

Následující kód vypočítá hodnotu kosinu určitého úhlu zadaného ve stupních. Ten je v následujícím příkladě představován proměnnou stupneUhlu.

```
int stupneUhlu = 45;
// Metoda Math.Cos vypočítá hodnotu kosinu úhlu zadaného v radiánech,
// pokud ho máme v stupních, musíme ho nejprve na radiány převést:
double uhelVRadianech = (Math.PI * stupneUhlu) / 180;
double cosUhlu = Math.Cos(uhelVRadianech);
Console.WriteLine("Kosinus úhlu {0} je {1}.", stupneUhlu, cosUhlu);
```

V tomto případě vypíše:

```
Kosinus úhlu 45 je 0,707106781186547.
```

80 Jak vypočítat tangens úhlu vyjádřeného ve stupních

Následující kód vypočítá hodnotu tangenty určitého úhlu zadaného ve stupních. Ten je v následujícím příkladě představován proměnnou stupneUhlu.

```
int stupneUhlu = 45;
// Metoda Math.Cos vypočítá hodnotu kosinu úhlu zadaného v radiánech,
// pokud ho máme v stupních, musíme ho nejprve na radiány převést:
double uhelVRadianech = (Math.PI * stupneUhlu) / 180;
double tanUhlu = Math.Tan(uhelVRadianech);
Console.WriteLine("Tangenta úhlu {0} je {1}.", stupneUhlu, tanUhlu);
```

V tomto případě vypíše:

```
Tangenta úhlu 45 je 1,61977519054386
```

81 Zaokrouhlení čísla na několik desetinných míst

Pokud chceme zaokrouhlit desetinné číslo na stanovený počet desetinných míst, použijeme metodu Round statické třídy System.Math. První argument této metody představuje

číslo, jež chceme zaokrouhlit, druhý pak počet desetinných míst, na které má být dané číslo zaokrouhleno.

```
double pi = 3.14159265;
double z = Math.Round(pi, 4);   // Vrátí 3.1416
```

82 Získání absoulutní hodnoty daného čísla

K získání absolutní hodnoty určitého čísla nám poslouží metoda Abs.

```
double VratAbsHodnotu(double cislo)
{
  if (cislo >= 0)
      return cislo;
  return (int)Math.Abs(cislo);
}
```

83 Výpočet n-té odmocniny daného čísla

Statická třída System.Math implicitně neobsahuje metodu pro výpočet n-té odmocniny z určitého čísla. Chceme-li ji vypočítat, umocníme pomocí metody Pow dané číslo převrácenou hodnotou čísla n.

```
double Odmocni(double cislo, double n)
{
  if (exp == 0)
     return 1;
  exp = 1 / exp;
  return Math.Pow(cislo, exp);
}
```

84 Výpočet n-té mocniny daného čísla

Pokud chceme umocnit číslo na určitý exponent, poslouží nám metoda System.Math.Pow. Tato metoda jako svůj první argument očekává číslo, které má být umocněno exponentem, jenž je představován druhým argumentem a vrací výsledek jako typ double.

```
double c = 2;
double exp = 3;
double vysledek = Math.Pow(c,exp);
```

85 Zaokrouhlení ceny na padesátihaléře

Následující metoda zaokrouhlí ceny v korunách s přesností na padesátihaléře. Pravidla, jež jsou stanovena Českou národní bankou, jsou pro zaokrouhlavání tato:

- Částka končící na 1 až 24 haléřů se zaokrouhlí na celé koruny dolů.
- Haléře u částky končící na 25 až 74 haléřů se zaokrouhlí na padesátihaléř.
- Částka končící na 75 až 99 haléřů se zaokrouhlí na celou korunu nahoru.

```
double ZaokrouhliCenu(double cena)
{
  double desCast = cena - Math.Floor(cena);
  if (desCast < 0.25f)
      return Math.Floor(cena);
  if (desCast < 0.75f)
      return Math.Floor(cena) + 0.5f;
  return (float)Math.Round(cena);
}
```

86 Výpočet logaritmu

Pokud chceme vypočítat logaritmus o daném základu, poslouží nám metoda Log třídy System.Math. Tato metoda očekává dva argumenty. Prvním z nich je číslo, jehož algoritmus chceme nalézt, druhým pak základ logaritmu.

```
int cislo = 55;
int zaklad = 4;
double log = Math.Log(cislo, zaklad);
```

V případě, že potřebujeme vypočítat dekadický logaritmus, použijeme metodu Math. Log10.

87 Převod hodnoty úhlu ve stupních na radiány

Následující výsek kódu převede hodnotu úhlu zadaného ve stupních na jednotky radiánu.

```
double uhel = 45;
double radiany = (uhel * Math.PI) / 180;
```

88 Vygenerování pseudonáhodného čísla

Pokud chceme vytvořit pseudonáhodné celé číslo, použijeme přetíženou metodu Next třídy System.Random, která jako argument očekává čísla typu int představující spodní a horní hranici zvýšenou o 1 intervalu čísel, jež mohou být vygenerována. Toto číslo by mělo být o 1 větší než námi požadovaná vrchní hranice intervalu.

Následující příklad vygeneruje číslo v intervalu 4–20:

```
Random r = new Random();
int spodniHranice = 4;
int horniHranice = 20;
double nahodneCislo = r.Next(spodniHranice,horniHranice + 1);
```

89 Vygenerování řetězce z náhodných znaků abecedy

Následující metoda vygeneruje řetězec z náhodných velkých písmen anglické abecedy, jejichž počet je definován číslem, které slouží jako její argument.

```
string GenerujZnaky(int pocet)
{
  StringBuilder vyslednyRetezec = new StringBuilder();
  int i = 0;
```

```
  Random r = new Random();
  while (i < pocet)
  {
    int m = r.Next(65, 91);
    char znak = (char)m;
    i++;
    vyslednyRetezec.Append(znak);
  }
  return vyslednyRetezec.ToString();
}
```

90 Vygenerování řetězce z náhodných písmen a číslic

Následující metoda vygeneruje řetězec náhodných znaků, jejichž počet je definován číslem sloužícím jako argument této metody. Samotné znaky jsou buď malá nebo velká písmena anglické abecedy nebo také číslice.

```
string GenerujZnaky(int pocet)
{
   string vyslednyRetezec = "";
   int i = 0;
   // Vytvoříme novou instanci třídy Random, pomocí které budeme
   // generovat náhodná čísla.
   Random r = new Random();
   while (i < pocet)
   {
     int m = 0;
     // Necháme náhodné číslo rozhodnout, zdali náhodně vybraným
     // znakem bude malé či velké písmeno nebo číslice.
     int l = r.Next(1, 4);
     switch (l)
     {
        case 1:
          m = r.Next(65, 91); break;
        case 2:
          m = r.Next(48, 58); break;
        case 3:
          m = r.Next(97, 122);break;
      }
      char znak = (char)m;
      i++;
      vyslednyRetezec += znak.ToString();
   }
   return vyslednyRetezec;
}
```

91 Náhodné vygenerování řetězce z předdefinovaných znaků

Následující metoda vygeneruje řetězec o určité délce, která je definována jejím prvním argumentem. Znaky, ze kterých bude řetězec náhodně poskládán, jsou uloženy v poli, které představuje argument druhý.

```
string VygenerujZnaky(int delka, char[] znaky)
{
  StringBuilder sb = new StringBuilder(delka);
  Random r = new Random();
```

```
  for (int x = 0; x < delka; x++)
  {
    int rand = r.Next(znaky.Length);
    sb.Append(znaky[rand]);
  }
  return sb.ToString();
}
```

92 Vygenerování série náhodných čísel

Následující metoda vrací pole celých čísel. Počet vygenerovaných čísel je stanoven prvním argumentem. Hranice spodního a vrchního intervalu, ve kterém bude náhodné číslo ležet, je definována zbylými dvěma argumenty.

```
int[] VygenerujCisla(int pocet, int min, int max)
{
  int[] nahCisla = new int[pocet];
  Random rand = new Random();
  for (int x = 0; x < pocet; x++)
  {
    nahCisla[x] =  (int)rand.Next(min, max+1);
  }
  return nahCisla;
}
```

93 Vygenerování pseudonáhodného čísla typu double v intervalu 0–1

Metoda NextDouble třídy System.Random vygeneruje pseudonáhodné desetinné číslo typu double, jež leží v intervalu <0-1), což znamená, že vygenerované číslo spadá do intervalu 0 až necelá 1. Jednotlivá čísla jsou v tomto intervalu rovnoměrně rozdělena, což znamená, že u každého čísla je stejná pravděpodobnost výběru.

```
Random r = new Random();
double nahCislo = r.NextDouble();
```

94 Ověření, zda jde o celé číslo

Chceme-li zjistit, zdali je určité číslo celé, zaokrouhlíme ho dolů metodou System.Math.Floor a následně odečteme od původního čísla. Je-li rozdíl roven 0, je toto číslo celé. K tomuto úkonu slouží následující metoda typu bool, která vrací true pouze v případě, že je číslo celé.

```
bool JeCele(double cislo)
{
  double desCast = Math.Floor(cislo) - cislo;
  if (desCast == 0)
     return true;
  return false;
}
```

95 Jak získat desetinnou část čísla

Chceme-li získat desetinnou část určitého čísla, nejprve zaokrouhlíme metodou Floor třídy System.Math toto číslo dolů a následně jej odečteme od původního čísla, čímž získáme jeho desetinnou část.

```
double cislo = 3.14159;
double desCast = cislo - Math.Floor(cislo);
Console.WriteLine("Desetinná část čísla {0} je {1}", cislo, desCast);
```

96 Řešení kvadratické rovnice

Následující metoda vyřeší kvadratickou rovnici, jsou-li zadány její koeficienty – ty jsou představovány argumenty této metody.

```
static string VypoctiKvadr(double a, double b, double c)
{
 double diskriminant = (b * b) - 4 * a * c;
 if (diskriminant < 0)
 {
   throw new ArgumentException
   ("Zadaná rovnice nemá v real. číslech řešení");
 }
 double x1 = (Math.Pow(diskriminant, 0.5) - b) / (2 * a);
 if (diskriminant == 0)
    return  x1.ToString();
 double x2 = (Math.Pow(diskriminant, 0.5) + b) / (2 * a);
 return x1.ToString() + " " + (x2 * -1).ToString() + "i";
}
```

97 Převedení časového údaje v sekundách na hodiny, minuty a sekundy

Chceme-li převést čas v sekundách na minuty, hodiny či dny, vytvoříme pomocí tovární metody FromSeconds novou instanci struktury TimeSpan, která bude obsahovat potřebné vlastnosti:

```
int sekundy = 36523;
TimeSpan ts = TimeSpan.FromSeconds(sekundy);

Console.WriteLine("{0} sekund je {1} dní, {2} hodin, {3} minut a {4}
sekund", sekundy, ts.Days,ts.Hours, ts.Minutes, ts.Seconds);
```

Pro vstup 36523 dostaneme následující výstup:

```
36523 sekund je 1 dní, 10 hodin, 8 minut a 43 sekund
```

98 Nalezení nejmenšího čísla v nesetříděném poli

Pokud chceme najít nejmenší číslo v nesetříděném poli, nejprve vytvoříme proměnnou nejmensiCislo typu int a uložíme do ní nejmenší hodnotu tohoto datového typu. Ta bude představována prvním prvkem v poli. Následně pole projdeme v cyklu a každé z čísel

porovnáme s proměnnou nejmensiCislo. Pokud je prvek pole menší, uložíme ho do této proměnné. Po ukončení všech iterací cyklu bude proměnná nejmensiCislo obsahovat nejmenší číslo v zadaném poli. Nakonec proměnnou nejmensiCislo vrátíme příkazem return.

```
int NajdiNejmensi(int[] cisla)
{
  int nejmensiCislo = cisla[0];
  foreach (int cislo in cisla)
  {
     if (cislo < nejmensiCislo)
     {
        nejmensiCislo = cislo;
     }
  }
  return nejmensiCislo;
}
```

99 Jak nalézt nejčastěji se vyskytující číslo v poli

Následující metoda vrátí číslo, jež se v poli vyskytuje nejčastěji. Pokud se počet výskytů několika různých čísel rovná, metoda vrátí to číslo, jehož první výskyt má v poli nejnižší index.

```
int NajdiNejcastejsiCislo(int[] cisla)
{
  int index = 0;
  int max = int.MinValue;
  // Postupně projdeme pole čísel a vždy vybereme jedno z nich.
  for (int x = 0; x < cisla.Length; x++)
  {
     int m = 0;
     // Projdeme všechna čísla, která v poli leží napravo od čísla
     // vybraného.
     for (int y = x + 1; y < cisla.Length; y++)
     {
       // Pokud se obě čísla rovnají, inkrementujeme proměnnou m
       // představující počet výskytů daného čísla v poli.
       if (cisla[x] == cisla[y])
           m++;
     }
     if (m > max)
     {
        max = m;
        index = x;
     }
  }
  return cisla[index];
}
```

100 Nalezení největšího čísla v poli čísel

Jestliže chceme najít nejmenší číslo v nesetříděném poli, vytvoříme nejprve proměnnou nejvetsiCislo typu int, do níž uložíme první prvek pole. Následně projdeme v cyklu všechny prvky pole a každý z nich porovnáme s proměnnou nejvetsiCislo. Je-li větší, uložíme

jeho hodnotu do proměnné nejvetsiCislo. Po ukončení všech iterací bude proměnná nejvetsiCislo obsahovat největší číslo v tomto poli.

```
int NajdiNejvetsiCislo(int[] cisla)
{
  int nejvetsiCislo = int.MinValue;
  foreach (int cislo in cisla)
  {
    if (cislo > nejvetsiCislo)
    {
       nejvetsiCislo = cislo;
    }
  }
  return nejvetsiCislo;
}
```

101 Zjištění souřadnic maxima kvadratické funkce

Následující metoda vypočítá hodnotu maxima paraboly kvadratické funkce z jejích koeficientů. Jednotlivé souřadnice maxima budou vypsány na konzolu. Proměnné a, b, c, které jsme deklarovali v hlavičce metody, představují jednotlivé koeficienty kvadratické funkce.

```
void NajdiVrcholy(double a, double b, double c)
{
   if (a == 0)
   {
     Console.WriteLine("Nejedná se o kvadratickou funkci");
   }
   else
   {
     double vrcholX0 = (b / (2 * a)) * -1;
     double vrcholY0 = c - ((b * b) / (4 * a));
     Console.WriteLine("Souřadnice x vrcholu rovnice je: " + vrcholX0);
     Console.WriteLine("Souřadnice y vrcholu rovnice je: " + vrcholY0);
   }
}
```

102 Jak v poli nalézt prvek a vrátit index jeho prvního či posledního výskytu

Jestliže chceme v poli vyhledat prvek a vrátit index jeho prvního výskytu, použijeme metodu IndexOf třídy System.Array. Jako její první argument dosadíme pole, ve kterém chceme prvek vyhledat. Ten je představován druhým argumentem. Jestliže tato metoda prvek v poli nenajde, vrátí -1.

```
int[] pole = {5, 8, -2, 41, 15};
int hledanyPrvek = 41;
int prvniVyskyt = pole.IndexOf(hledanyPrvek);  // Vrátí 3
```

Chceme-li vrátit index posledního výskytu prvku v poli, použijeme metodu LastIndexOf.

103 Jak vypočítat aritmetický průměr z hodnot uložených v poli

Chceme-li spočítat aritmetický průměr hodnot, jež jsou uloženy v poli, sečteme je v cyklu a součet uložíme do proměnné hodnota. Tu pak vydělíme počtem prvků daného pole. Počet prvků tu reprezentuje vlastnost Length.

```
double SpoctiArPrumer(int[] cisla)
{
  double hodnota = 0;
  foreach (int c in cisla)
  {
     hodnota += c;
  }
  hodnota /= cisla.Length;
  return hodnota;
}
```

104 Jak vypočítat geometrický průměr z hodnot uložených v poli

Následující metoda spočítá geometrický průměr z hodnot, jež jsou uloženy v poli, které je použito jako její argument.

```
double SpoctiGeoPrumer(int[] cisla)
{
   double hodnota = 1;
   foreach (int c in cisla)
   {
      hodnota *= c;
   }
   hodnota = Math.Pow(hodnota,(1 /(double)cisla.Length));
   return hodnota;
}
```

105 Výpočet faktoriálu

Následující metoda spočítá faktoriál čísla, které je jejím argumentem. Pokud je toto číslo záporné nebo větší než 20, metoda vyvolá výjimku.

```
long Faktorial(int n)
{
   if ( n < 0)
       throw new ArgumentException("Zadané číslo musí být kladné !");
   if (n > 20)
       throw new ArgumentException("Číslo nesmí být větší než 20");
   long vysledek = 1;
   for (int x = 2; x <= n; x++)
   {
     vysledek *= x;
   }
   return vysledek;
}
```

106 Výpočet počtu variací n prvků k-té třídy bez opakování

Následující metoda vypočítá počet variací z n prvků k-té třídy bez opakování.

```
long Variace(int n, int k)
{
   long pocetVariaci = 1;
   for (int x =0; x < k; x++)
   {
      pocetVariaci *= n-x;
   }
   return pocetVariaci;
}
```

107 Vypočet počtu kombinací z n prvků k-té třídy bez opakování

Následující metoda vrací číslo představující počet kombinací z n prvků k-té třídy.

```
long SpocitejKombinace(int k, int n)
{
  int a = n + k - 1;
  return Faktorial(a) / (Faktorial(k) * (Faktorial(n - 1));
}
```

108 Výpočet počtu kombinací z n prvků k-té třídy s opakováním

Následující metoda vrací číslo představující počet kombinací z n prvků k-té třídy s opakováním.

```
long SpocitejKombinace(int k, int n)
{
  if ((k + n) >= 21)
     throw new ArgumentException("n + k musí být menší než 21!");
  int a = n + k - 1;
  int c = n - 1;
  return Faktorial(a) / (Faktorial(k) * Faktorial(c));
}
```

109 Jak zjistit trojúhelníkovou nerovnost

Následující metoda očekává jako argumenty jednotlivé strany trojúhelníka a vrátí hodnotu typu bool v závislosti na tom, zda trojúhelník lze setrojit, či nikoli. Sestrojit ho lze právě tehdy, jestliže je součet dvou různých stran trojúhelníka větší než délka strany třetí. Tímto způsobem prověříme všechny tři strany. Pokud lze trojúhelník sestrojit, vrátí tato metoda true, v opačném případě vrátí false.

```
bool LzeSestrojit(int a, int b, int c)
{
  if (a + b < c)
```

```
      return false;
   if (a + c < b)
      return false;
   if (b + c < a)
      return false;
   return true;
}
```

110 Zjištění pohlaví z rodného čísla

V rodném čísle mají ženy k datu měsíce narození přičteno 50, a proto, chceme-li zjistit kterému z pohlaví náleží dané rodné číslo, vyjmeme ze řetězce podřetězec představující měsíc a převedeme jej na číslo. Pokud je toto číslo větší než 12, datum patří ženě. V opačném případě muži.

```
string rodneCislo = "xxxxxx/xxxx";
if (int.Parse(rodneCislo.Substring(2, 2)) > 12)
{
   Console.WriteLine("Toto rodné číslo patří ženě");
   return;
}
Console.WriteLine("Toto rodné číslo patří muži");
```

111 Ověření validity rodného čísla

Pokud chceme zjistit, zda je validní rodné číslo, které bylo vydáno po roce 1953, spočítáme zbytek po vydělení prvních devíti čísel rodného čísla číslem 11. Je-li roven poslední číslici (pro 10 je to 0), rodné číslo je validní (tedy je to možné rodné číslo).

```
bool OverRodneCislo(string rodneCislo)
{
  rodneCislo = rodneCislo.Replace("/", "");
  int n = int.Parse(rodneCislo.Substring(0, 9));
  int posledniCifra = int.Parse(rodneCislo.Substring(9, 1));
  int zbytekPoDeleni = n % 11;
  if ((zbytekPoDeleni == 10 && zbytekPoDeleni == posledniCifra) ||
  zbytekPoDeleni == posledniCifra)
     return true;
  return false;
}
```

112 Ověření validity IČO

Jestliže chceme ověřit správnost IČO (identifikační číslo organizace), vynásobíme jeho prvních sedm cifer čísly od 8 do 2 a jednotlivé výsledky postupně sečteme. Poté z výsledného součtu vypočítáme zbytek po dělení číslem 11, a pokud je zbytek po dělení:

- roven 1, poslední, tedy osmá cifra IČO, by se měla rovnat 0.
- roven 0, poslední cifra IČO by se měla rovnat 1.
- roven 10, poslední cifra IČO by se měla rovnat také 1.
- ve zbylých případech platí: poslední cifra = 11 – zbytek po dělení.

Následující metoda vrací hodnotu typu bool v závislosti na tom, zda je IČO validní. Pokud ano, metoda vrací true, v opačném případě však false.

```
bool OverICO(string cislo)
{
  int m = 0;
  if (cislo.Length != 8)
  {
    Console.WriteLine("Číslo musí mít osm cifer");
    return false;
  }
  for (int x = 0; x < cislo.Length - 1; x++)
  {
    int c = Convert.ToInt32(cislo[x].ToString());
    m += c * (8 - x);
  }
  int zbytek = m % 11;
  int poslCislo = Convert.ToInt32(cislo[7].ToString());
  if ((zbytek == 0 || zbytek == 10) && poslCislo == 1)
     return true;
  if (zbytek == 1 && poslCislo == 0)
     return true;
  if (poslCislo == 11 - zbytek)
      return true;
  return false;
}
```

113 Vypočet poslední (kontrolní) číslice IČO


pokročilý

Následující metoda očekává jako argument číslo představující prvních sedm číslic IČO. Metoda spočítá osmou (kontrolní) číslici a vrátí celé IČO.

```
int VygenerujICO(int cis)
{
  string cislo = cis.ToString();
  int m = 0;
  for (int x = 7; x > 0; x--)
  {
    int c = int.Parse(cislo[7-x].ToString());
    m += c * (x + 1);
  }
  int zbytek = int.Parse(cislo) % 11;
  if (zbytek == 0 || zbytek == 10)
     return int.Parse(cislo.ToString() + "1");
  else if (zbytek == 1)
     return int.Parse(cislo.ToString() + "0");
  else
     return int.Parse(cislo + (11 - zbytek).ToString());
}
```

114 Převod čísla z decimální soustavy do binární


pokročilý

Následující metoda převede číslo zapsané v desítkové soustavě do soustavy binární.

```
string PrevedDoBinarni(int cislo)
{
  string bin = "";
```

```
  while (cislo != 0)
  {
    bin += (cislo % 2).ToString();
    cislo /= 2;
  }
  string s = "";
  for (int x = bin.Length - 1; x >= 0; x--)
  {
    s += bin[x].ToString();
  }
  return s;
}
```

115 Převod čísla ze šestnáctkové soustavy

Následující metoda převede číslo v hexadecimální soustavy do soustavy desítkové.

```
int HexToDec(string hexCislo)
{
  int vysledek = 0;
  int exp = 0;
  foreach(char c in hexCislo.ToCharArray())
  {
     int n;
     if (char.IsDigit(c))
        n = Convert.ToInt32(c.ToString());
     else
        n = (int)c-55;
     vysledek += n * (int)Math.Pow(16, hexCislo.Length - exp-1);
     exp++;
   }
   return vysledek;
}
```

116 Převod čísla o základu menším než 10 na číslo typu int

Následující metoda převede řetězec představující zápis čísla o základu menším než 10 na číslo typu int.

```
static int Převeď(string číslo, int základ)
{
   int výsledek = 0;
   int pom = 1;
   for (int x = číslo.Length - 1; x >= 0; x--)
   {
       výsledek += pom*Int32.Parse(číslo[x].ToString());
       pom *= základ;
   }
   return výsledek;
}
```

117 Převod čísla typu int na číslo v rozmezí 2–16

Následující metoda převede všechny hodnoty typu int (kromě int.MinValue) na číslo o základu v rozmezí 2–16.

```
static string DoSoustavy(int číslo, int základ)
{
   if (číslo == 0)
      return "0"; // Je-li to nula, není co řešit.
   string číslice = "0123456789ABCDEF";
   Stack<char> obráceně = new Stack<char>();
   bool záporné = číslo < 0;
   if (záporné) číslo = -číslo;
   while (číslo != 0)
   {
     obráceně.Push(číslice[číslo % základ]);
     číslo /= základ;
   }
   if (záporné) obráceně.Push('-');
   // Získal jsem číslice v obráceném pořadí.
   // Otočím je pomocí zásobníku. Je to rychlejší než
   // vkládat na počátek pomocí metody Insert.
   StringBuilder hex = new StringBuilder();
   foreach (char c in obráceně)
          hex.Append(c);
   return hex.ToString();
}
```

118 Jak rozložit číslo na součin prvočísel


pokročilý

Následující metoda rozloží číslo na součin prvočísel, která následně vypíše na konzolu. Pokud dané číslo rozložit nelze, metoda vypíše příslušné hlášení.

```
void RozlozCislo(int cislo)
{
  bool jePrvocislo = true;
  StringBuilder vysledek = new StringBuilder(cislo.ToString() + " = ");
  for (int x = 2; x <= cislo; x++)
  {
     while (cislo % x == 0)
     {
       jePrvocislo = false;
       vysledek.Append( x.ToString() + "*");
       cislo = cislo / x;
     }
  }
  if (jePrvocislo)
  Console.WriteLine("Zadané číslo nelze rozložit na součin prvočísel");
  else
    Console.WriteLine(vysledek.ToString().TrimEnd('*'));
}
```

Zadáme-li jako vstup například číslo 224, dostaneme následující výstup:

```
224 = 2*2*2*2*2*7
```

119 Seřazení řetězců od nejkratšího po nejdelší


znalec

Chceme-li setřídit pole řetězců dle jejich délky, použijeme metodu Sort třídy System.Array a jako její první argument dosadíme pole řetězců, jež chceme setřídit. Druhý argument bude představován instancí třídy implementující rozhraní System.Collections.IComparer.

Tato třída musí obsahovat veřejnou metodu Compare očekávající dva argumenty typu object představující porovnávané objekty. V našem případě chceme porovnávat délku řetězců, a proto objekty, jež představují argument této metody, přetypujeme na typ string a následně pomocí vlastnosti Length získáme jejich délku. Pokud je první z nich delší, vrátíme 1. Je-li tomu naopak, tak -1. Jsou-li oba stejné, vrátíme 0. Takto by mohla vypadat deklarace komparátoru:

```
class Komparator : IComparer
{
  public int Compare(object x, object y)
  {
    string a = (string)x;
    string b = (string)y;
    if (a.Length > b.Length)
       return 1;
    if (b.Length > a.Length)
       return -1;
    return 0;
  }
}
```

Pokud zavoláme tuto funkci následujícím kódem,

```
public static void Main()
{
 string[] retezce = {"myš","dlouhý řetězec","trouba","práce"};
 Array.Sort(retezce, new Komparator());
 foreach (string retezec in retezce)
 {
   Console.WriteLine(retezec);
 }
}
```

dostaneme následující výstup:

```
myš
práce
trouba
dlouhý řetězec
```

120 Součet prvních n členů aritmetické posloupnosti

Následující kód vypočítá součet všech členů, které leží v intervalu od prvního po n-tý člen aritmeticky uspořádané posloupnosti dle vzorce

hodnota n-tého členu = první hodnota + diference * n

```
// Proměnná min představuje hodnotu prvního prvku posloupnosti.
int min = 0;
// Proměnná d představuje rozdíl mezi jednotlivými prvky posloupnosti.
int d = 2;
// Proměnná n představuje pozici n-tého prvku v posloupnosti, jehož
// hodnotu chceme vypočítat.
int n = 20;
int soucet = min + d * n;
```

121 Součet prvních n členů geometrické posloupnosti

Následující kód vypočítá součet všech členů, které leží v intervalu od prvního po n-tý člen geometrické posloupnosti.

```
int SpoctiSoucinClenu(int prvniClen, double q,int n)
{
   if (q != 1)
      return prvniClen * (Math.Pow(q,n)-1) / (q-1));
   return prvniClen * n;
}
```

122 Zjištění, zda je posloupnost klesající, konstantní či rostoucí

Následující metoda zjistí, zda čísla uložená v poli tvoří rostoucí, klesající nebo konstantní posloupnost či žádnou z nich.

```
void ZjistiRustPosloupnosti(double[] hodnoty)
{
  if (hodnoty.Length < 2)
  {
     Console.WriteLine("Posloupnost musí obsahovat alespoň dva prvky");
     return;
  }
  double rozdil = hodnoty[0] - hodnoty[1];
  for (int x = 1; x < hodnoty.Length; x++)
  {
     double h = hodnoty[x - 1] - hodnoty[x];
     if (Math.Sign(rozdil) != Math.Sign(h))
     {
        Console.WriteLine("Nelze určit typ růstu posloupnosti.");
        break;
     }
  }
  if (rozdil < 0)
      Console.WriteLine("Posloupnost je rostoucí");
  else if (rozdil > 0)
      Console.WriteLine("Posloupnost je klesající");
  else
      Console.WriteLine("Posloupnost je konsntantní");
}
```

123 Tvoří čísla uložená v poli aritmetickou posloupnost?

Následující metoda vrací hodnotu typu bool v závislosti na tom, zda čísla poli, jež je argumentem této metody, tvoří aritmetickou posloupnost. Pokud ano, metoda vrací true, jinak vrací false.

```
bool JeLinearni(double[] cisla)
{
  if (cisla.Length < 3)
  {
    throw new Exception("Pole musí obsahovat nejméně tři prvky");
```

```
  }
  double rozdil = cisla[0] - cisla[1];
  for (int x = 2; x < cisla.Length; x++)
  {
      if (!(cisla[x - 1] - cisla[x] == rozdil))
          return false;
  }
  return true;
}
```

124 Tvoří čísla uložená v poli geometrickou posloupnost?


pokročilý

Následující metoda vrací hodnotu typu bool v závislosti na tom, zda čísla v poli, jež je argumentem této metody, tvoří geometrickou posloupnost. Pokud ano, metoda vrací true, jinak vrací false.

```
bool JeGeometricka(double[] pole)
{
  // Pokud pole obsahuje méně než tři prvky, vyvoláme výjimku.
  if (pole.Length < 3)
  {
    throw new Exception("Pole musí obsahovat minimálně tři prvky");
  }
  // Spočítáme podíl, který by měl být stejný pro všechny prvky, které
  // leží v poli vedle sebe tak, aby jednotlivá čísla pole tvořila
  // geometrickou posloupnost.
  double podil = pole[1] / pole[0];
  // Postupně projdeme jednotlivé prvky - čísla v poli.
  // Pokud není podíl dvou vedle sebe ležících čísel roven proměnné
  // podíl, tato posloupnost není geometrická, čili
  // vrátíme false.
  for (int x = 2; x < pole.Length; x++)
  {
    if (!(pole[x] / pole[x - 1] == podil))
        return false;
  }
  return true;
}
```

125 Vypsání všech čísel dělitelných daným číslem v určitém intervalu


začátečník

Následující metoda vypíše všechna čísla, jež leží v určitém intervalu a jsou dělitelná daným číslem.

```
void VypisDelence(int min, int max, int delitel)
{
   int m = 0;
   for (int x = min; x < min + delitel; x++)
   {
      if (x % delitel == 0)
      {
          m = x; break;
      }
   }
```

```
    for (int x = m; x <= max; x = x+ delitel)
    {
       Console.WriteLine(x);
    }
}
```

126 Výpočet n-té číslice Fibonacciho posloupnosti

Následující metoda vypočte n-tý prvek Fibonacciho posloupnosti.

```
public static int Fibonacci(int index)
{
    if(index == 0) return 0;
    if(index == 1) return 1;
    int prePre = 0;
    int pre = 1;
    int vysledek = 0;
    for(int i = 1; i < index; i++)
    {
        vysledek = prePre + pre;
        prePre = pre;
        pre = vysledek;
    }
    return vysledek;
}
```

127 Výpis písmen anglické abecedy

Následující algoritmus vypíše všechna velká písmena anglické abecedy.

```
for (int i = 65; i <= 90; i++)
{
    Console.WriteLine((char)i);
}
```

128 Jsou čísla v poli setříděna od největšího po nejmenší?

Pokud chceme zjistit, zda je číselné pole setříděné, projdeme jeho jednotlivé prvky, a pokud je každé číslo $i_x > i_x+1$, je pole setříděno od největšího po nejmenší.

```
bool JeSetridene(int[] pole)
{
  for (int x = 1; x < pole.Length; x++)
  {
     if (pole[x - 1] < pole[x])
         return false;
  }
  return true;
}
```

129 Jsou čísla v poli setříděna od nejmenšího po největší?

Pokud chceme zjistit, zda je číselné pole setříděné, projdeme jeho jednotlivé prvky, a pokud je každé číslo ix < ix+1, tak je pole setříděno od nejmenšího čísla po největší.

```
bool JeSetridene(int[] pole)
{
  for (int x = 1; x < pole.Length; x++)
  {
    if (pole[x - 1] > pole[x])
       return false;
  }
  return true;
}
```

130 Jak setřídit čísla uložená v poli

Chceme-li setřídit pole čísel, použijeme metodu Sort třídy System.Array, která toto pole očekává jako svůj argument.

```
int[] cisla = { 5, 1, 55, -3, 6 };
Array.Sort(cisla);
```

Ještě bych rád dodal, že metoda Sort umožňuje třídění prvků jakéhokoli typu. Jejich typ však musí implementovat rozhraní System.Collections.IComparer.

131 Jak v setříděném poli vyhledat prvek

Jestliže chceme zjistit index určitého prvku v setříděném poli, použijeme metodu BinarySearch třídy System.Array, která jako svůj argument očekává pole, ve kterém chceme daný prvek vyhledat. Ten je představován druhým argumentem. Pokud hledaný prvek v poli obsažen není, metoda vrací -1.

```
string[] setridenePole = {"hledaný prvek","něco", "něco jiného"};
string hledanyPrvek = "hledaný prvek";
int index = Array.BinarySearch(setridenePole, hledanyPrvek); // Vrátí 0
```

132 Převrácení pořadí znaků v řetězci

Chceme-li získat řetězec obsahující znaky jiného řetězce, ale seřazené pozpátku, nejprve tento řetězec převedeme metodou ToCharArray na pole znaků, jejichž pořadí v poli převrátíme statickou metodou Reverse třídy Array. Nakonec toto pole použijeme jako argument konstruktoru třídy String, čímž získáme výsledný řetězec.

```
static string PřevraťŘetězec(string řetězec)
{
   char[] znaky = řetězec.ToCharArray();
   Array.Reverse(znaky);
   return new String(znaky);
}
```

Pro vstup:

```
string retezec = "Žluťoučký kůň příšerně úpěl ďábelské ódy";
Console.WriteLine(PrevraťŘetězec(retezec));
```

Dostaneme následující text

```
ydó ékslebáď lěpú ěnrešířp ňůk ýkčuoťulŽ
```

133 Jak vypočítat medián

Chceme-li v nesetříděném poli zjistit medián, nejprve je setřídíme metodou System.Array.Sort a následně vybereme prvek, který leží přesně uprostřed. Index prostředního prvku získáme tak, že počet prvků pole (vlastnost Length) vydělíme dvěma. Následně vrátíme číslo, jež leží na této pozici.

```
int[] pole = { 8, 5, -3, -7, 22, 12,1 };
Array.Sort(pole);
int prostredniPrvek = pole[pole.Length / 2];
Console.WriteLine(prostredniPrvek);     // Vypíše 5.
```

134 Jak nalézt modus ve skupině čísel

Následující metoda najde v poli čísel číslo představující modus, tedy číslo, které je v ní obsaženo nejčastěji.

```
double NajdiModus(double[] cisla)
{
  // Pole setřídíme:
  Array.Sort(cisla);
  double d = -1;
  int i =0;
  int j = 0;
  bool g = false;
  for (int x = 1; x < cisla.Length; x++)
  {
     if (cisla[x] == cisla[x - 1])
     {
        g = true;
        i++;
     }
     else if (g)
     {
          if (i > j)
          {
             j = i;
             d = cisla[x - 1];
          }
          i = 0;
          g = false;
     }
  }
  return d;
}
```

135 Jak vypočítat průměrnou odchylku

Metoda vypočítá průměrnou odchylku z hodnot, které jsou uloženy v poli předaném jako argument.

```
double PrumernaOdchylka(double[] hodnoty)
{
  double prumer = 0;
  for (int x = 0; x < hodnoty.Length; x++)
```

```
        prumer += hodnoty[x];
  prumer /= hodnoty.Length;
  double vysledek = 0;
  for (int x = 0; x < hodnoty.Length; x++)
  {
       vysledek += Math.Abs(prumer - hodnoty[x]);
   }
   vysledek /= hodnoty.Length;
   return vysledek;
}
```

136 Jak vypočítat rozptyl

Metoda vypočítá rozptyl z hodnot, které jsou uloženy v poli předaném jako argument.

```
double Rozptyl(double[] hodnoty)
{
  double s1 = 0;
  double s2 = 1;

  foreach (double d in hodnoty)
  {
     s1 += d;
     s2 *= d;
  }
  double rozptyl = (s2-(s1 * s1 / hodnoty.Length))/(hodnoty.Length - 1);
  return rozptyl;
}
```

137 Jak vypočítat směrodatnou odchylku

Chceme-li vypočítat směrodatnou odchylku z určité skupiny hodnot, vypočítáme nejprve jejich rozptyl (metodu počítající rozptyl jsem uvedl v předchozím tipu) a ten následně odmocníme metodou Math.Pow.

```
double rozptyl = Rozptyl(hodnoty);
double odchylka = Math.Sqrt(rozptyl);
```

138 Jak spočítat řetězový index

Následující metoda spočítá řetězový index z hodnot, jež jsou uloženy v poli předaném jako argument. Vypočtené hodnoty vrátí v poli typu double.

```
double[] IndexRetezovy(double[] hodnotyObdobi)
{
  double[] m = new double[hodnotyObdobi .Length];
  for (int x = 1; x < p.Length; x++)
  {
     m[x] = hodnotyObdobi[x] / hodnotyObdobi[x - 1];
  }
  return m;
}
```

139 Jak spočítat základní index

Následující metoda spočítá základní index z hodnot, jež jsou uloženy v poli předaném jako argument. Vypočtené hodnoty budou vráceny v poli typu double.

```
double[] IndexZakladni(double[] p)
{
    double[] m = new double[p.Length];
    for (int x = 0; x < p.Length; x++)
    {
       m[x] = p[x] / p[0];
    }
    return m;
}
```

140 Jak spočítat tempo přírůstku

Následující metoda spočítá tempo přírůstku oproti předchozím obdobím z hodnot, jež jsou uloženy v poli předaném jako argument. Jednotlivá tempa přírustku budou vrácena v poli typu double.

```
double[] TempoPrirustku(double[] p)
{
   double[] m = new double[p.Length];
   for (int x = 1; x < p.Length; x++)
   {
       m[x] = (p[x] - p[x - 1]) / p[x - 1];
   }
   return m;
}
```