

**Gymnázium, Praha 10, Voděradská 2**

**Projekt LITERACY**

**Sbírka složitějších příkladů pro výuku programování v jazyce C#**

* **předmět IVT**

**Výukový materiál vytvořený v rámci projektu LITERACY**

Autor: Mgr. Petr Laštovička Ing. Markéta Kožíšková

Předmět: IVT

Datum: 27. 11. 2013

Praha & EU: Investujeme do vaší budoucnosti Evropský sociální fond

Projekt LITERACY financovaný v rámci Operačního programu Praha – Adaptabilita

Obsah

[1. Základní algoritmické konstrukce, cyklus a rekurze 3](#_Toc373308418)

[1.1. Větvení SWITCH, větvení IF - ELSE 3](#_Toc373308419)

[1.2. Cyklus FOR, WHILE 4](#_Toc373308420)

[1.3. Použití rekurzivního algoritmu 6](#_Toc373308421)

[2. Strukturované datové typy 8](#_Toc373308422)

[2.1. Naplnění a výpis pole 8](#_Toc373308423)

[2.2. Pole s proměnnou délkou. Napište program pro výpočet průměru zadaných čísel. 8](#_Toc373308424)

[2.3. Stringové pole a příkaz Sort (program JmenaTrideni). 8](#_Toc373308425)

[2.4. Stringové pole a příkaz IndexOf (program JmenaOblibeny). 8](#_Toc373308426)

[2.5. Pole ve formulářích 9](#_Toc373308427)

[2.6. List – základní operace 11](#_Toc373308428)

[2.7. Listbox – základní operace 12](#_Toc373308429)

[3. Práce s textem a soubory 13](#_Toc373308430)

[3.1. Práce se znakem a řetězci 13](#_Toc373308431)

[3.2. Práce s textovým souborem 17](#_Toc373308432)

[3.3. Kódování znaků 18](#_Toc373308433)

[4. Algoritmy vnitřního třídění 19](#_Toc373308434)

[4.1. SelectionSort 19](#_Toc373308435)

[4.2. InsertionSort 21](#_Toc373308436)

[4.3. BubbleSort 22](#_Toc373308437)

[4.4. QuickSort 23](#_Toc373308438)

[5. Numerické algoritmy 24](#_Toc373308439)

[6. Uživatelské rozhraní 27](#_Toc373308440)

[7. Objektově orientované programování 29](#_Toc373308441)

[8. Grafika 32](#_Toc373308442)

# 1. Základní algoritmické konstrukce, cyklus a rekurze

## 1.1. Větvení SWITCH, větvení IF - ELSE

-napište program kalkulačka, který umožní na základě uživatelské volby vybrat jednu z aritmetických operací

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Vítejte v kalkulačce");

string pokracovat = "ano";

while (pokracovat == "ano")

{

Console.WriteLine("Zadejte první číslo:");

float a = float.Parse(Console.ReadLine());

Console.WriteLine("Zadejte druhé číslo:");

float b = float.Parse(Console.ReadLine());

Console.WriteLine("Zvolte si operaci:");

Console.WriteLine("1 - sčítání");

Console.WriteLine("2 - odčítání");

Console.WriteLine("3 - násobení");

Console.WriteLine("4 - dělení");

int volba = int.Parse(Console.ReadLine());

float vysledek = 0;

switch (volba)

{

case 1:

vysledek = a + b;

break;

case 2:

vysledek = a - b;

break;

case 3:

vysledek = a \* b;

break;

case 4:

vysledek = a / b;

break;

}

if ((volba > 0) && (volba < 5))

Console.WriteLine("Výsledek: {0}", vysledek);

else

Console.WriteLine("Neplatná volba");

Console.WriteLine("Přejete si zadat další příklad? [ano/ne]");

pokracovat = Console.ReadLine();

}

Console.WriteLine("Děkuji za použití kalkulačky, aplikaci ukončíte libovolnou klávesou.");

Console.ReadKey();

}

## 1.2. Cyklus FOR, WHILE

-napište program, který zjistí, zda zadané číslo je prvočíslo. Algoritmus řešení navrhněte tak, aby program pomocí FOR cyklu testoval podmínku if ((cislo % delitel) == 0)

Při každém opakování inkrementujte proměnnou delitel.

static void Main(string[] args)

{

Console.WriteLine("zadej cislo");

int cislo = Convert.ToInt32(Console.ReadLine());

int JeDelitelne = 0;

for (int delitel = 2; delitel < cislo; delitel++) // kazdy z tech 1000 cisel otestujem, jestli jsou delitelny vsema moznejma mensima cislama

{

// znaminko % znamena zbytek po deleni

if ((cislo % delitel) == 0) // pokud je cislo delitelny delitelem, tak to prvocislo neni

{

JeDelitelne++;

}

}

if (JeDelitelne == 0)

{ Console.WriteLine("Je to prvocislo"); }

else

{ Console.WriteLine("Neni to Prvocislo"); }

Console.ReadKey();

}

**-Matematicky lepší způsob**

static void Main(string[] args)

{

Console.WriteLine("Vítejte, vložte číslo, které chcete podrobit zkoušce.");

int cislo = Convert.ToInt32(Console.ReadLine());

bool takCoBude = true;

for (int i =2; i <= Math.Sqrt(cislo); i++)

{

if (cislo % i == 0)

{

Console.WriteLine("Zadané číslo není prvočíslo, protože ho dělí číslo " + i.ToString());

takCoBude = false;

}

}

if (takCoBude == true)

{

Console.WriteLine("Zadané číslo je prvočíslo.");

}

Console.ReadKey();

}

-napište program, který zjistí, zda zadané číslo je prvočíslo. Pokud není, vypíšou se všichni jeho dělitelé a nejbližší nižší prvočíslo.

static void Main(string[] args)

{

Console.Write("zadej nejake cislo: ");

int cislo = Convert.ToInt32(Console.ReadLine());

Console.WriteLine();

int PocetDelitelu = 0;

//zjištění počtu dělitelů

//----------------------------------------------------

for (int delitel = 2; delitel < cislo; delitel++)

{

// znaminko % znamena zbytek po deleni

if ((cislo % delitel) == 0)

{

PocetDelitelu++;

Console.WriteLine(PocetDelitelu +". Delitel: " + delitel + " ;");

}

}

//------------------------------------------------------

// otestování podmínky o počtu dělitelů čísla

if (PocetDelitelu == 0)

Console.WriteLine("Cislo " + cislo + " je prvocislo");

else

{ Console.WriteLine("Cislo " + cislo + " neni prvocislo"); }

//hledání nejbližšího nižšího prvočísla

//----------------------------------------------------

do

{

cislo--;

int delitel = cislo-1;

PocetDelitelu = 0;

while (delitel > 1)

{

if ((cislo % delitel) == 0)

{

++PocetDelitelu;

}

delitel--;

}

}

while (PocetDelitelu > 0) ;

//----------------------------------------------------

Console.WriteLine("Nejblizsi nizsi prvocislo je: "+cislo);

Console.ReadKey();

}

## 1.3. Použití rekurzivního algoritmu

-Napište program pro výpočet faktoriálu čísla za užití rekurzivního postupu

class ProgramFactorial

{

private static int rekurze;

static void Main(string[] args)

{

int x;

rekurze = 0;

Console.WriteLine("Zadejte číslo: ");

x = int.Parse(Console.ReadLine());

Console.WriteLine("Faktorial cisla "+ x + " je " +FaktorialRekurzi(x));

Console.WriteLine("Metoda rekurze byla použita: "+rekurze+ "x");

Console.ReadLine();

}

public static int FaktorialRekurzi(int x)

{

rekurze++;

if (x < 2)

{ return 1; }

else

{

return x \* FaktorialRekurzi(x - 1);

}

}

}

-Napište program pro výpis x členů Fibonachiho posloupnosti za užití rekurzivního postupu i bez něj.

namespace FibonachiRek

{

class Program

{

private static int rekurze;

static void Main(string[] args)

{

int pocet;

//ošetření vstupu - povolíme pouze nezáporná čísla

do

{

Console.Write("Kolikáté Fibonacciho číslo si přeješ?: ");

pocet = int.Parse(Console.ReadLine());

} while (pocet < 0);

//výpočet pomocí první (rekurzivní) metody

rekurze = 0;

Console.WriteLine(pocet + ". číslo Fibonacciho posloupnosti je: " +

FibonacciRek(pocet));

Console.WriteLine("Pro výpočet bylo potřeba " + rekurze +

" zavolání výpočetní metody.");

//výpočet pomocí druhé (nerekurzivní) metody

rekurze = 0;

Console.WriteLine(pocet + ". číslo Fibonacciho posloupnosti je: " +

FibonacciNerek(pocet));

Console.WriteLine("Pro výpočet bylo potřeba " + rekurze +

" iterací výpočetního cyklu.");

Console.ReadLine();

}

//rekurzivní metoda pro výpočet x-tého Fibonacciho čísla

//tato metoda je velmi pomalá, protože potřebuje mnoho rekurzivních

//volání (spuštění)

//- a tento počet roste v závislosti na x exponenciálně!

public static int FibonacciRek(int x)

{

rekurze++; //zvýšíme hodnotu proměnné rekurze o jedničku

int vysledek;

if (x <= 1) //nulté a první fib. č.

{

vysledek = x;

}

else

{

vysledek = FibonacciRek(x - 1) + FibonacciRek(x - 2);

}

return vysledek;

}

//nerekurzivní varianta metody pro výpočet x-tého Fibonacciho čísla

//v proměnných "a" a "b" máme zadány hodnoty prvních dvou prvků

//další počítáme průchodem od nuly po "x"

//tato metoda má lieární složitost - cyklus vždy proběhne (x-1) krát

public static int FibonacciNerek(int x)

{

int vysledek;

if (x <= 1) //nulté a první fib. č.

{

{

vysledek = x;

}

}

else

{

int a = 0; //nulté fib. č., dále pak představuje fib. č. (n-2)

int b = 1; //první fib. č., dále pak představuje fib. č. (n-1)

int c = 0; //pomocná proměnná, představuje fib. č. (n)

for (int n = 1; n < x; n++)

{

rekurze++;

c = b + a; //fib(n) = fib(n-1) + fib(n-2)

//pro další iteraci uděláme z ...

a = b; //...fib(n-1) -> fib(n-2)

b = c; //...fib(n) -> fib(n-1)

//... a pak se cyklus opakuje, dokud je to potřeba

}

vysledek = c;

}

return vysledek;

}

}

}

# 2. Strukturované datové typy

## 2.1. Naplnění a výpis pole

int[] pole = new int[10];

pole[0] = 1;

for (int i = 0; i < 10; i++)

pole[i] = i + 1;

// foreach (int i in pole)

// Console.Write("{0} ", i);

for (int i = 0; i < pole.Length; i++)

Console.Write("{0} ", pole[i]);

Console.ReadKey();

## 2.2. Pole s proměnnou délkou. Napište program pro výpočet průměru zadaných čísel.

Console.WriteLine("Ahoj, spočítám ti průměr čísel. Kolik čísel zadáš?");

int pocet = int.Parse(Console.ReadLine());

int[] cisla = new int[pocet];

for (int i = 0; i < pocet; i++)

{

Console.Write("Zadejte {0}. číslo: ", i + 1);

cisla[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Průměr tvých čísel je: {0}", cisla.Average());

Console.ReadKey();

## 2.3. Stringové pole a příkaz Sort (program JmenaTrideni).

Naplňte pole jmény a setřiďte je podle abecedy.

Console.WriteLine("Kolik zadáš jmen?");

int pocet = int.Parse(Console.ReadLine());

string[] SeznamJmen = new string[pocet];

for (int i = 0; i < pocet; i++)

{

Console.Write("Zadejte {0}. jméno: ", i + 1);

SeznamJmen[i] = (Console.ReadLine());

}

Array.Sort(SeznamJmen);

foreach (string s in SeznamJmen)

Console.Write("{0} ", s);

Console.ReadKey();

## 2.4. Stringové pole a příkaz IndexOf (program JmenaOblibeny).

Najděte pořadí textového řetězce ve stringovém poli.

Console.WriteLine("Kolik zadáš jmen?");

int pocet = int.Parse(Console.ReadLine());

string[] SeznamJmen = new string[pocet];

for (int i = 0; i < pocet; i++)

{

Console.Write("Zadejte {0}. jméno: ", i + 1);

SeznamJmen[i] = (Console.ReadLine());

}

Console.WriteLine("Ahoj, zadej svého oblíbeného Spolužáka: ");

string Jmeno = Console.ReadLine();

int pozice = Array.IndexOf(SeznamJmen, Jmeno);

if (pozice >= 0)

Console.WriteLine("Jo, to je můj {0}. nejoblíbenější spolužák!", pozice + 1);

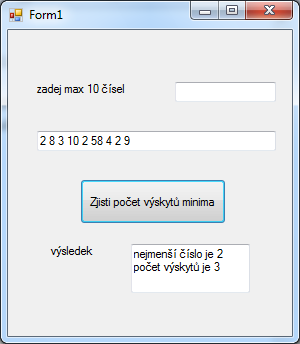
else

Console.WriteLine("Hele, tohle není není z naší třídy!");

Console.ReadKey();

## 2.5. Pole ve formulářích

Vytvořte pole maximálně 10 čísel.

Čísla zadávejte přes textbox, při vkládání oddělujte entrem.

Při načítání do pole čísla současně souhrnně ukládejte do jiného textboxu, čísla oddělujte mezerami.

Zjistěte nejmenší číslo v poli a kolikrát se v řadě vyskytuje. Výsledek vytiskněte do dalšího textového pole.

namespace uloha\_poleminimum

{

public partial class Form1 : Form

{

int[] polecisel = new int[10];

int index = 0;

int min;

int pocetmin;

public Form1()

{

InitializeComponent();

textBoxPole.Text = "";

textBoxVstup.Text = "";

textBoxVysledek.Text = "";

}

private void textBoxVstup\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

polecisel[index] = Convert.ToInt32(textBoxVstup.Text);

textBoxPole.Text += textBoxVstup.Text + " ";

index++;

textBoxVstup.Text = "";

}

}

private void tlacitkoZjisti\_Click(object sender, EventArgs e)

{

min = polecisel[0];

for (int i = 1; i < index; i++)

{

if (polecisel[i] < min)

{

min = polecisel[i];

}

}

textBoxVysledek.Text = "nejmenší číslo je " + min.ToString() +

Environment.NewLine;

pocetmin = 0;

for (int i = 0; i < index; i++)

{

if (polecisel[i] == min)

pocetmin++;

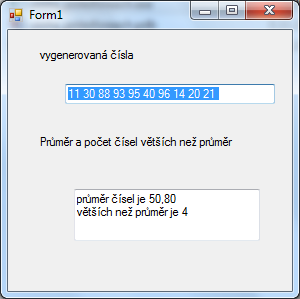
}

textBoxVysledek.Text += "počet výskytů je " + pocetmin.ToString();

}

}

}

Vytvořte 10 prvkové pole a naplňte ho náhodnými čísly z intervalu <0,100>

Vygenerovaná čísla současně kontrolně vypište do textového pole, oddělujte mezerami.

Zjistěte průměr čísel a kolik z nich je větší než průměr. Pro následné zpracování pole použijte (kde je to možné) cyklus foreach.

Program bude řídit událost Load okna (formuláře).

namespace uloha\_polesforeach

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

textBoxVstup.Text = "";

textBoxVystup.Text = "";

}

private void Form1\_Load(object sender, EventArgs e)

{

int[] polecisel = new int[10];

Random rnd = new Random();

for (int i = 0; i < 10; i++)

{

polecisel[i] = rnd.Next(101);

textBoxVstup.Text += polecisel[i] + " ";

}

double prumer;

double soucet = 0;

foreach (int polozka in polecisel)

{

soucet += polozka;

}

prumer = soucet / polecisel.Length;

textBoxVystup.Text = "průměr čísel je " + prumer.ToString("F2") +

Environment.NewLine;

int pocetvetsich = 0;

foreach (int polozka in polecisel)

{

if (polozka > prumer)

pocetvetsich++;

}

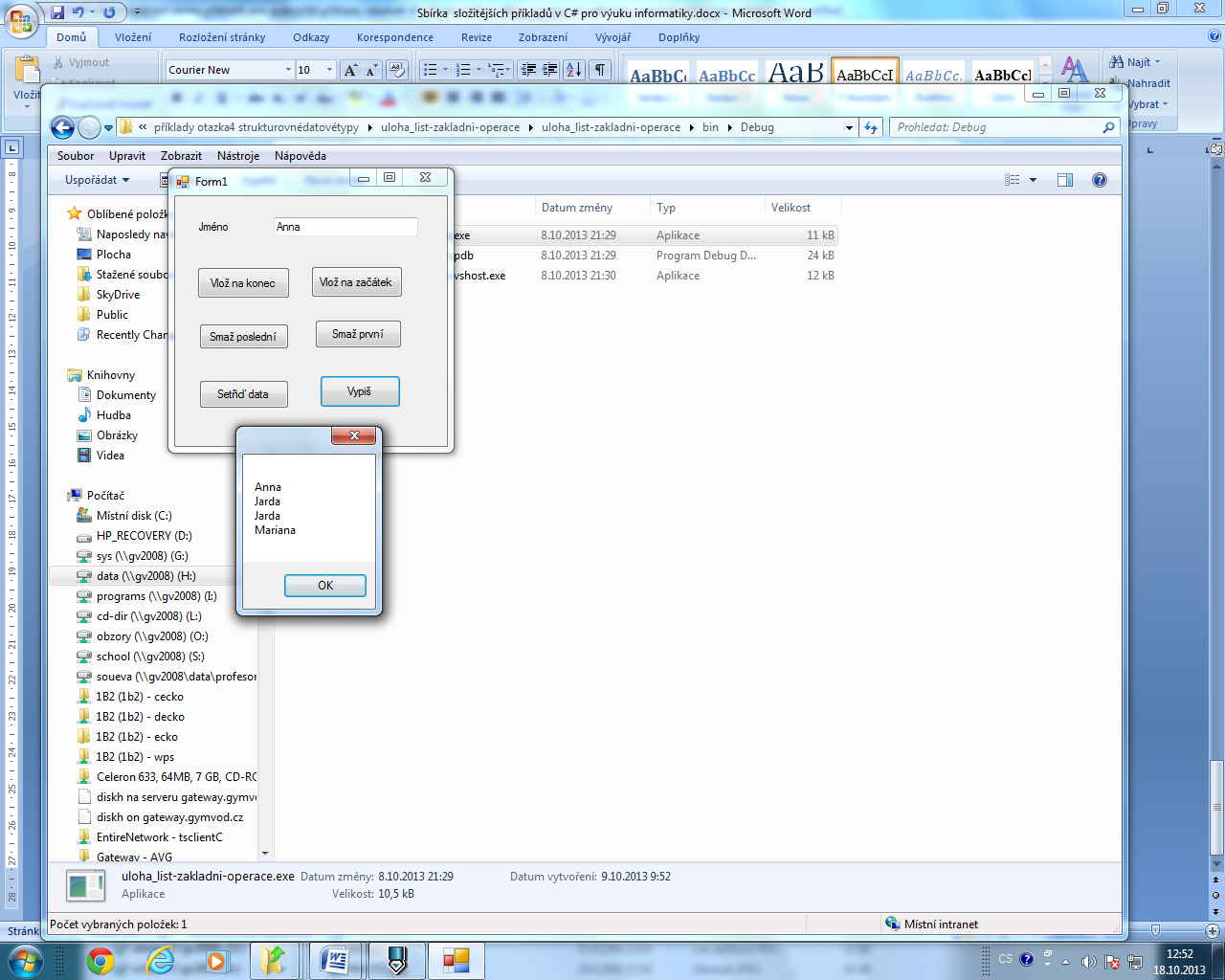
textBoxVystup.Text += "větších než průměr je " + pocetvetsich.ToString();

}

}

}

## 2.6. List – základní operace

Vytvořte program s grafickým rozhraním s tlačítky

umožňující realizovat základní operace s datovým typem List:

- vložení jména zadaného přes textové pole na konec Listu

- vložení jména zadaného přes textové pole na začátek Listu

- zrušení posledního prvku

- zrušení prvního prvku

- abecední setřídění prvků

- výpis odřádkovaného obsahu listu do Messageboxu

namespace uloha\_list\_zakladni\_operace

{

public partial class Form1 : Form

{

List<string> polejmen = new List<string>();

public Form1()

{

InitializeComponent();

}

private void buttonVlozNaKonec\_Click(object sender, EventArgs e)

{

polejmen.Add(textBoxJmeno.Text);

}

private void buttonVlozNaZacatek\_Click(object sender, EventArgs e)

{

polejmen.Insert(0, textBoxJmeno.Text);

}

private void buttonSmazPosledni\_Click(object sender, EventArgs e)

{

polejmen.RemoveAt(polejmen.Count - 1);

}

private void buttonSmazPrvni\_Click(object sender, EventArgs e)

{

polejmen.RemoveAt(0);

}

private void buttonSetrid\_Click(object sender, EventArgs e)

{

polejmen.Sort();

}

private void buttonVypis\_Click(object sender, EventArgs e)

{

string vystup = "";

foreach (string polozka in polejmen)

{

vystup += polozka + Environment.NewLine;

}

MessageBox.Show(vystup);

}

}

}

## 2.7. Listbox – základní operace

Vytvořte program s grafickým rozhraním s tlačítky

umožňující realizovat základní operace s prvkem listbox:

- vložení jména zadaného přes textové pole na konec listboxu

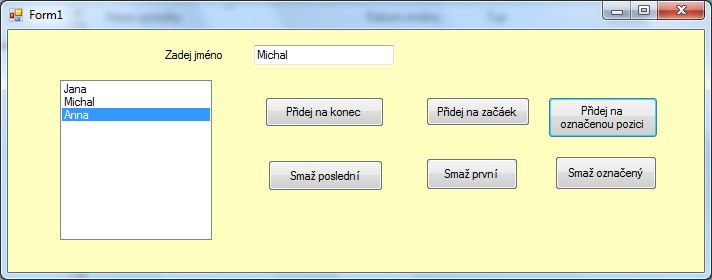
- vložení jména zadaného přes textové pole na začátek listboxu

- vložení jména zadaného přes textové pole na zadanou pozici

- zrušení posledního prvku

- zrušení prvního prvku

- zrušení označeného prvku



namespace uloha\_listbox\_zakladni\_operace

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void buttonPridejNaKonec\_Click(object sender, EventArgs e)

{

listBoxJmena.Items.Add(textBoxJmeno.Text);

}

private void buttonPridejNaZacatek\_Click(object sender, EventArgs e)

{

listBoxJmena.Items.Insert(0, textBoxJmeno.Text);

}

private void buttonPridejNaPozici\_Click(object sender, EventArgs e)

{

int index = listBoxJmena.SelectedIndex;

//musíme ošetřit situaci, kdy není vybrána žádná pozice

if (index >= 0) listBoxJmena.Items.Insert(index, textBoxJmeno.Text);

}

private void buttonSmazPoslední\_Click(object sender, EventArgs e)

{

listBoxJmena.Items.RemoveAt(listBoxJmena.Items.Count - 1);

}

private void buttonSmazPrvni\_Click(object sender, EventArgs e)

{

listBoxJmena.Items.RemoveAt(0);

}

private void buttonSmazVybrany\_Click(object sender, EventArgs e)

{

int index = listBoxJmena.SelectedIndex;

if (index >= 0) listBoxJmena.Items.RemoveAt(index);

}

}

}

# 3. Práce s textem a soubory

## 3.1. Práce se znakem a řetězci

Připravte program na základní metody pro práci se stringem - operace realizujte vždy stiskem tlačítka:

1, Převeďte zadaný řetězec na velká písmena

2, Odstraňte ze zadaného řetězce mezery

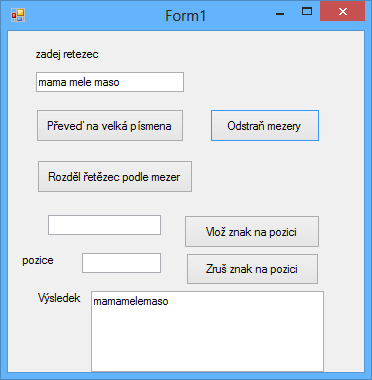
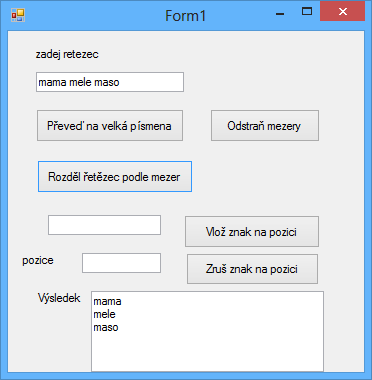
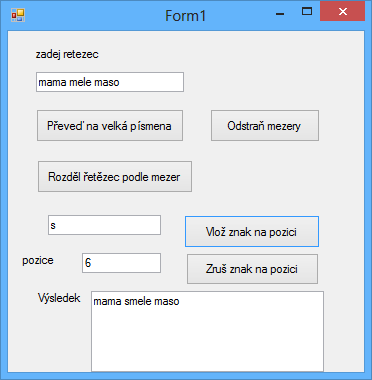
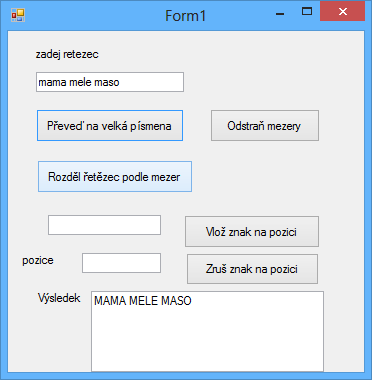
- počáteční a koncové mezery

- mezery mezi slovy

3, Rozdělte řetězec na více podřetězců, jako oddělovač zvolte mezeru

4, Vložte do řetězce znak a to na zadanou pozici

5, Odstraňte z řetězce znak na zadané pozici



namespace uloha\_retezce

{

public partial class Form1 : Form

{

string retezec1;

string retezec2;

string vysledek;

public Form1()

{

InitializeComponent();

}

private void buttonToUpper\_Click(object sender, EventArgs e)

{

retezec1 = textBoxRetezec1.Text;

vysledek = retezec1.ToUpper();

textBoxVysledek.Text = vysledek;

}

private void buttonOdstranenimezer\_Click(object sender, EventArgs e)

{

retezec1 = textBoxRetezec1.Text;

//odstraní počáteční a koncové mezery

vysledek = retezec1.Trim();

//varianta 1 - nahrazení mezer prázdným řetězcem

vysledek = vysledek.Replace(" ", string.Empty);

//varianta 2 - odstranění mezer, resp. znaku na dané pozici

//pozor string je read-only, vždy nutno uložit přiřazovacím příkazem

/\*for (int index=0;index<vysledek.Length;index++)

{

if (vysledek[index] == ' ')

//parametry metody Remove (pozice, počet odstraňovaných znaků)

vysledek = vysledek.Remove(index, 1);

}\*/

textBoxVysledek.Text = vysledek;

}

private void buttonRozdelRetezec\_Click(object sender, EventArgs e)

{

textBoxVysledek.Text = "";

string[] poleretezcu = new string[10];

retezec1 = textBoxRetezec1.Text;

poleretezcu = retezec1.Split(' ');

for (int index = 0; index < poleretezcu.Length; index++)

{

string polozka = poleretezcu[index];

if (polozka != "")

textBoxVysledek.Text += polozka + Environment.NewLine;

}

}

private void buttonVlozZnak\_Click(object sender, EventArgs e)

{

retezec1 = textBoxRetezec1.Text;

retezec2 = textBoxRetezec2.Text;

int pozice = Convert.ToInt32(textBoxPozice.Text);

//uvažujeme, že uživatel označuje pozice od 1

vysledek = retezec1.Insert(pozice - 1, retezec2);

textBoxVysledek.Text = vysledek;

}

private void buttonSmazZnak\_Click(object sender, EventArgs e)

{

retezec1 = textBoxRetezec1.Text;

int pozice = Convert.ToInt32(textBoxPozice.Text);

//uvažujeme, že uživatel označuje pozice od 1

vysledek = retezec1.Remove(pozice - 1, 1);

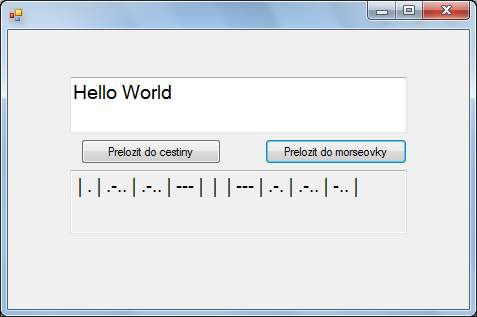
textBoxVysledek.Text = vysledek;

}

}

}

Napište program, který převede text do znaků Morseovy abecedy a zpět. Abecední znaky definujte jako řetězec znaků a pro jeho robytí použijte metodu SPLIT.



namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void doMorseovky\_Click(object sender, EventArgs e)

{

{

string s = textBox1.Text;

string zprava = "";

string abecedniZnaky = "abcdefghijklmnopqrstuvwxyz";

char[] abeceda = abecedniZnaky.ToCharArray();

string[] morseovyZnaky = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....",

"..", ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", "...", "-", "..-",

"...-", ".--", "-..-", "-.--", "--.."};

char[] znaky = s.ToCharArray();

foreach (char abecedniznak in znaky)

{

string morseuvZnak = "";

int index = Array.IndexOf(abeceda, abecedniznak);

if (index >= 0) // znak nalezen

morseuvZnak = morseovyZnaky[index];

zprava += (morseuvZnak + " | ");

}

textBox3.Text = zprava;

}

}

private void doCestiny\_Click(object sender, EventArgs e)

{

string s = textBox1.Text;

string zprava = "";

string abecedniZnaky = "abcdefghijklmnopqrstuvwxyz";

string[] morseovyZnaky = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....",

"..", ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", "...", "-", "..-",

"...-", ".--", "-..-", "-.--", "--.."};

string[] znaky = s.Split(' ');

foreach (string morseuvZnak in znaky)

{

char abecedniZnak = '@';

int index = Array.IndexOf(morseovyZnaky, morseuvZnak);

if (index >= 0) // znak nalezen

abecedniZnak = abecedniZnaky[index];

zprava += abecedniZnak;

textBox3.Text = zprava;

}

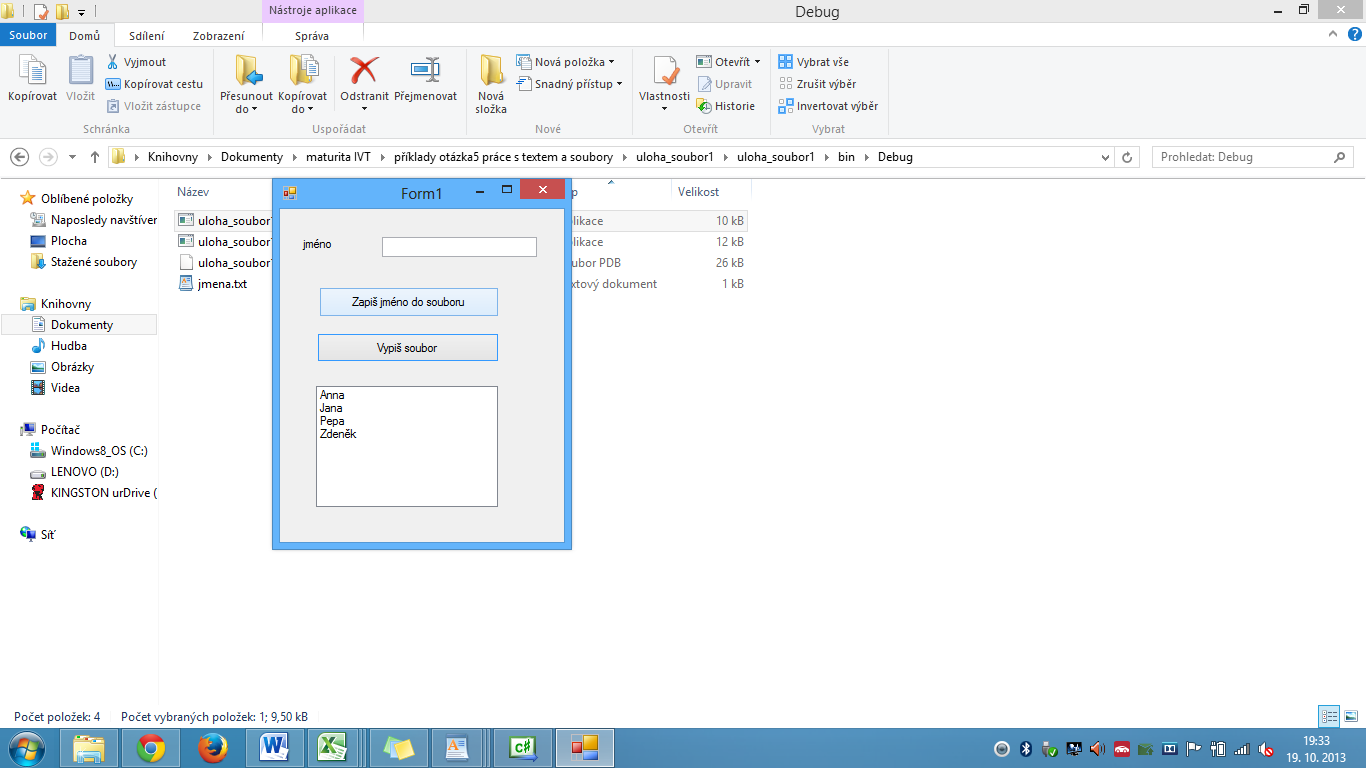
}

}

}

## 3.2. Práce s textovým souborem

Vytvořte textový soubor obsahující jména zadávaná postupně uživatelem do textového pole. Následně obsah souboru vypište do listboxu.



// odkaz na jmenný prostor obsahující třídy pro práci se soubory

using System.IO;

namespace uloha\_soubor1

{

public partial class Form1 : Form

{

//inicializace souborů

StreamWriter souborjmena1;

StreamReader souborjmena2;

public Form1()

{

InitializeComponent();

}

//nutno na začátku vymazat obsah souboru

private void Form1\_Load(object sender, EventArgs e)

{

souborjmena1 = new StreamWriter("jmena.txt");

souborjmena1.Write("");

souborjmena1.Close();

}

private void buttonZapisDoSouboru\_Click(object sender, EventArgs e)

{

souborjmena1 = new StreamWriter("jmena.txt", true);

//parametr true říká, že další položky se přidávají na konec souboru

souborjmena1.WriteLine(textBoxJmeno.Text);

textBoxJmeno.Text = "";

souborjmena1.Close();

}

private void buttonVypisRadekSouboru\_Click(object sender, EventArgs e)

{

souborjmena2 = new StreamReader("jmena.txt");

string radek = souborjmena2.ReadLine();

while (radek != null)

{

listBoxJmena.Items.Add(radek);

radek = souborjmena2.ReadLine();

}

souborjmena2.Close();

}

}

}

## 3.3. Kódování znaků

a, Vypište všechna velká písmena abecedy – výstup do MessageBoxu

b, Vypisujte náhodně velká písmena abecedy – výstup do TextBoxu

Při řešení využijte ASCII tabulku.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace uloha\_kodovaniznaku

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

textBoxVysledek.Text = "";

}

private void buttonVypisAbecedu\_Click(object sender, EventArgs e)

{

//písmeno A má kód 65, Z 90

string vysledek = "";

char pismeno;

for (int i = 65; i <= 90; i++)

{

pismeno = (char)i;

vysledek += pismeno + " ";

}

MessageBox.Show(vysledek);

}

private void buttontiskninahodne\_Click(object sender, EventArgs e)

{

//generujeme náhodná čísla v rozsahu podle ascii tabulky, převedeme na písmena

Random rnd = new Random();

int nahcislo = rnd.Next(65, 91);

char pismeno = (char)nahcislo;

textBoxVysledek.Text += pismeno + " ";

}

}

}

# 4. Algoritmy vnitřního třídění

## 4.1. SelectionSort

-Napište program, který naplní pole čísly a setřídí hodnoty v poli vzestupně pomocí algoritmu Select Sort.

namespace SelectionSort

{

class Program

{

static void Main(string[] args)

{

// naplnění pole čísel

Console.WriteLine("Kolik čísel zadáš?");

int pocet = int.Parse(Console.ReadLine());

int[] PoleCisel = new int[pocet];

for (int i = 0; i < PoleCisel.Length; i++)

{

Console.Write("Zadejte {0}. číslo: ", i + 1);

PoleCisel[i] = int.Parse(Console.ReadLine());

}

// výpis neseřazeného pole

Console.WriteLine("Pole před použitím Selection Sortu : ");

for (int i = 0; i < pocet; i++)

{

Console.Write(" " +PoleCisel[i]);

}

Console.WriteLine();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Teď použijeme Selection Sort

int temp, minimum;

int prohoz = 0;

for (int j = 0; j < pocet - 1; j++)

{

minimum = j;

for (int k = j + 1; k < pocet; k++)

{

if (PoleCisel[k] < PoleCisel[minimum])

{

minimum = k;

}

}

temp = PoleCisel[minimum];

PoleCisel[minimum] = PoleCisel[j];

PoleCisel[j] = temp;

prohoz++;

}

// konec řadícího algoritmu

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// výpis seřazeného pole

Console.WriteLine("Pole po seřazení Selection Sortem: ");

for (int i = 0; i < pocet; i++)

{

Console.Write(" " + PoleCisel[i]);

}

Console.WriteLine("Bylo použito " +prohoz+" prohození");

Console.ReadKey();

}

}

}

-lepší zápis

namespace SelectionSort

{

class Program

{

public static void selectionSort(int[] pole)

{

int temp, min;

int prohoz = 0;

for (int i = 0; i < (pole.Length-1); i++)

{

min = i;

// hledani minima

for (int j = i; j < (pole.Length); j++)

{

if (pole[min] > pole[j])

min = j;

// prohozeni prvku

temp = pole[min];

pole[min] = pole[i];

pole[i] = temp;

prohoz++;

}

}

Console.Write("Bylo použito prohození: {0} ", prohoz);

}

static void Main(string[] args)

{

Console.WriteLine("Kolik čísel zadáš?");

int pocet = int.Parse(Console.ReadLine());

int[] cisla = new int[pocet];

for (int i = 0; i < pocet; i++)

{

Console.Write("Zadejte {0}. číslo: ", i + 1);

cisla[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Seřazené pole: ");

selectionSort(cisla);

for (int i = 0; i < pocet; i++)

Console.Write(" {0}",cisla[i]);

Console.ReadKey();

Console.ReadKey();

}

}

}

## 4.2. InsertionSort

-Napište program, který naplní pole čísly a setřídí hodnoty v poli vzestupně pomocí algoritmu Insert Sort.

namespace InsertionSort

{

class Program

{

//metoda řazení InsertSort

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public static void insertionSort(int[] pole)

{

int prvek, j;

int presun = 0;

for (int i = 1; i <= (pole.Length - 1); i++)

{

// ulozeni prvku

prvek = pole[i];

j = i - 1;

while ((j >= 0) && (pole[j] > prvek))

{

pole[j + 1] = pole[j];

j--;

presun++;

}

pole[j + 1] = prvek;

}

Console.WriteLine("Bylo použito presunuti: {0} ", presun);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static void Main(string[] args)

{

Console.WriteLine("Kolik čísel zadáš?");

int pocet = int.Parse(Console.ReadLine());

int[] cisla = new int[pocet];

for (int i = 0; i < pocet; i++)

{

Console.Write("Zadejte {0}. číslo: ", i + 1);

cisla[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Seřazené pole: ");

insertionSort(cisla);

for (int i = 0; i < pocet; i++)

Console.Write(" {0}", cisla[i]);

Console.ReadKey();

Console.ReadKey();

}

}

}

## 4.3. BubbleSort

-Napište program, který naplní pole čísly a setřídí hodnoty v poli vzestupně pomocí algoritmu Bubble Sort.

namespace BubbleSort

{

class Program

{

//metoda řazení BubbleSort

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public static void bubbleSort(int[] pole)

{

int j = pole.Length - 2, temp;

int PocetProhozeni = 0;

// kontrola prohozeni

bool prohoz = true;

while (prohoz)

{

prohoz = false;

for (int i = 0; i <= j; i++)

{

// prohozeni

if (pole[i] > pole[i + 1])

{

temp = pole[i];

pole[i] = pole[i + 1];

pole[i + 1] = temp;

prohoz = true;

PocetProhozeni++;

}

}

j--;

}

Console.WriteLine("Bylo použito presunuti: {0} ", PocetProhozeni);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static void Main(string[] args)

{

Console.WriteLine("Kolik čísel zadáš?");

int pocet = int.Parse(Console.ReadLine());

int[] cisla = new int[pocet];

for (int i = 0; i < pocet; i++)

{

Console.Write("Zadejte {0}. číslo: ", i + 1);

cisla[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Seřazené pole: ");

bubbleSort(cisla);

for (int i = 0; i < pocet; i++)

Console.Write(" {0}", cisla[i]);

Console.ReadKey();

Console.ReadKey();

}

}

}

## 4.4. QuickSort

-Napište program, který naplní pole čísly a setřídí hodnoty v poli vzestupně pomocí algoritmu Quick Sort.

namespace QuickSort

{

class Program

{

// preusporada pole na prvky mensi nez pivot, pivot a prvky vetsi nez pivot

public static int rozdel(int[] pole, int vlevo, int vpravo, int pivot)

{

int temp = pole[pivot]; // prohozeni pivotu s poslednim prvkem

pole[pivot] = pole[vpravo];

pole[vpravo] = temp;

int i = vlevo;

for (int j = vlevo; j < vpravo; j++)

{

if (pole[j] < pole[vpravo])

{ // prvek je mensi, nez pivot

temp = pole[i]; // prohozeni pivotu s prvkem na pozici

pole[i] = pole[j];

pole[j] = temp;

i++; // posun pozice

}

}

temp = pole[i]; // prohozeni pivotu zpet

pole[i] = pole[vpravo];

pole[vpravo] = temp;

return i; // vrati novy index pivotu

}

public static void omezenyQuicksort(int[] pole, int vlevo, int vpravo)

{

if (vpravo >= vlevo)

{ // podminka rekurze

int pivot = vlevo; // vyber pivotu

int novyPivot = rozdel(pole, vlevo, vpravo, pivot);

// rekurzivni zavolani na obe casti pole

omezenyQuicksort(pole, vlevo, novyPivot - 1);

omezenyQuicksort(pole, novyPivot + 1, vpravo);

}

}

// zavola omezeny quicksort na cele pole

public static void quicksort(int[] pole)

{

omezenyQuicksort(pole, 0, pole.Length - 1);

}

static void Main(string[] args)

{

Console.WriteLine("Kolik čísel zadáš?");

int pocet = int.Parse(Console.ReadLine());

int[] cisla = new int[pocet];

for (int i = 0; i < pocet; i++)

{

Console.Write("Zadejte {0}. číslo: ", i + 1);

cisla[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Seřazené pole: ");

quicksort(cisla);

for (int i = 0; i < pocet; i++)

Console.Write(" {0}", cisla[i]);

Console.ReadKey();

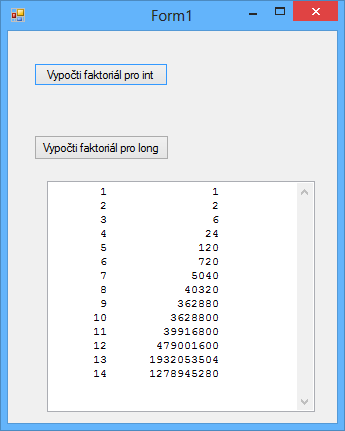
Console.ReadKey();

}

}

}

# 5. Numerické algoritmy

Zjistěte, pro jak velké číslo lze vypočítat faktoriál, aniž dojde k přetečení:

a, pro datový typ int

b, pro datový typ long

Výsledky výpočtu vytiskněte přehledně ve sloupcích zarovnaných zprava.

Pro úspěšné ladění programu je nutno mít v nastavení projektu (Project - Build - Advanced) odškrtnuto Check for arithmetic overflow.

namespace uloha\_faktorial

{

public partial class Form1 : Form

{

string vysledek;

public Form1()

{

InitializeComponent();

}

private void tlacitkoVypoctiInt\_Click(object sender, EventArgs e)

{

int n = 1;

int faktorials = 1;

int faktorialn = 1;

vysledek = n.ToString().PadLeft(8) + faktorials.ToString().PadLeft(16) + Environment.NewLine;

textBoxVysledek.Text = vysledek;

n = 2;

do

{

faktorials = faktorialn;

faktorialn = 1;

for (int i = n; i >= 1; i--)

{

faktorialn \*= i;

}

vysledek = n.ToString().PadLeft(8) + faktorialn.ToString().PadLeft(16) + Environment.NewLine;

textBoxVysledek.Text += vysledek;

n++;

}

while (faktorials < faktorialn);

}

private void tlacitkoVypoctiLong\_Click(object sender, EventArgs e)

{

int n = 1;

long faktorials = 1;

long faktorialn = 1;

vysledek = n.ToString().PadLeft(8) + faktorials.ToString().PadLeft(25) + Environment.NewLine;

textBoxVysledek.Text = vysledek;

n = 2;

do

{

faktorials = faktorialn;

faktorialn = 1;

for (int i = n; i >= 1; i--)

{

faktorialn \*= i;

}

vysledek = n.ToString().PadLeft(8) + faktorialn.ToString().PadLeft(25) + Environment.NewLine;

textBoxVysledek.Text += vysledek;

n++;

}

while (faktorials < faktorialn);

}

}

}

Př. Vyvtořte program, který bude převádět

1, zadané desítkové číslo do soustavy binární

2, zadané desítkové číslo do soustavy hexadecimální

Při převodu na písmena využijte konstrukci switch-case.

namespace uloha\_prevody\_soustav

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void buttonNaDvojkove\_Click(object sender, EventArgs e)

{

int cislo = Convert.ToInt32(textBoxDesitkove.Text);

int cislopodeleni = cislo;

int zbytek;

string vysledek = "";

string cislice;

string vsechnycislice;

while (cislopodeleni != 1)

{

//zjistíme zbytek po dělení

zbytek = cislopodeleni % 2;

//zjisíme celočíselný podíl

cislo = cislopodeleni;

cislopodeleni = cislo / 2;

cislice = zbytek.ToString();

// nuly a jedničky ukládáme do výsledného stringu zleva

vsechnycislice = vysledek.Insert(0, cislice);

vysledek = vsechnycislice;

}

// převádíme koncovou jedničku

vsechnycislice = vysledek.Insert(0, "1");

vysledek = vsechnycislice;

textBoxDvojkove.Text = vysledek;

}

private void buttonNaHexadecimalni\_Click(object sender, EventArgs e)

{

int cislo = Convert.ToInt32(textBoxDesitkove.Text);

int cislopodeleni = cislo;

int zbytek;

string vysledek = "";

string cislice;

string vsechnycislice;

while (cislopodeleni > 15)

{

zbytek = cislopodeleni % 16;

cislo = cislopodeleni;

cislopodeleni = cislo / 16;

cislice = zbytek.ToString();

switch (cislice)

{

case "10": cislice = "A";

break;

case "11": cislice = "B";

break;

case "12": cislice = "C";

break;

case "13": cislice = "D";

break;

case "14": cislice = "E";

break;

case "15": cislice = "F";

break;

default:

break;

}

vsechnycislice = vysledek.Insert(0, cislice);

vysledek = vsechnycislice;

}

cislice = cislopodeleni.ToString();

switch (cislice)

{

case "10": cislice = "A";

break;

case "11": cislice = "B";

break;

case "12": cislice = "C";

break;

case "13": cislice = "D";

break;

case "14": cislice = "E";

break;

case "15": cislice = "F";

break;

default:

break;

}

vsechnycislice = vysledek.Insert(0, cislice);

vysledek = vsechnycislice;

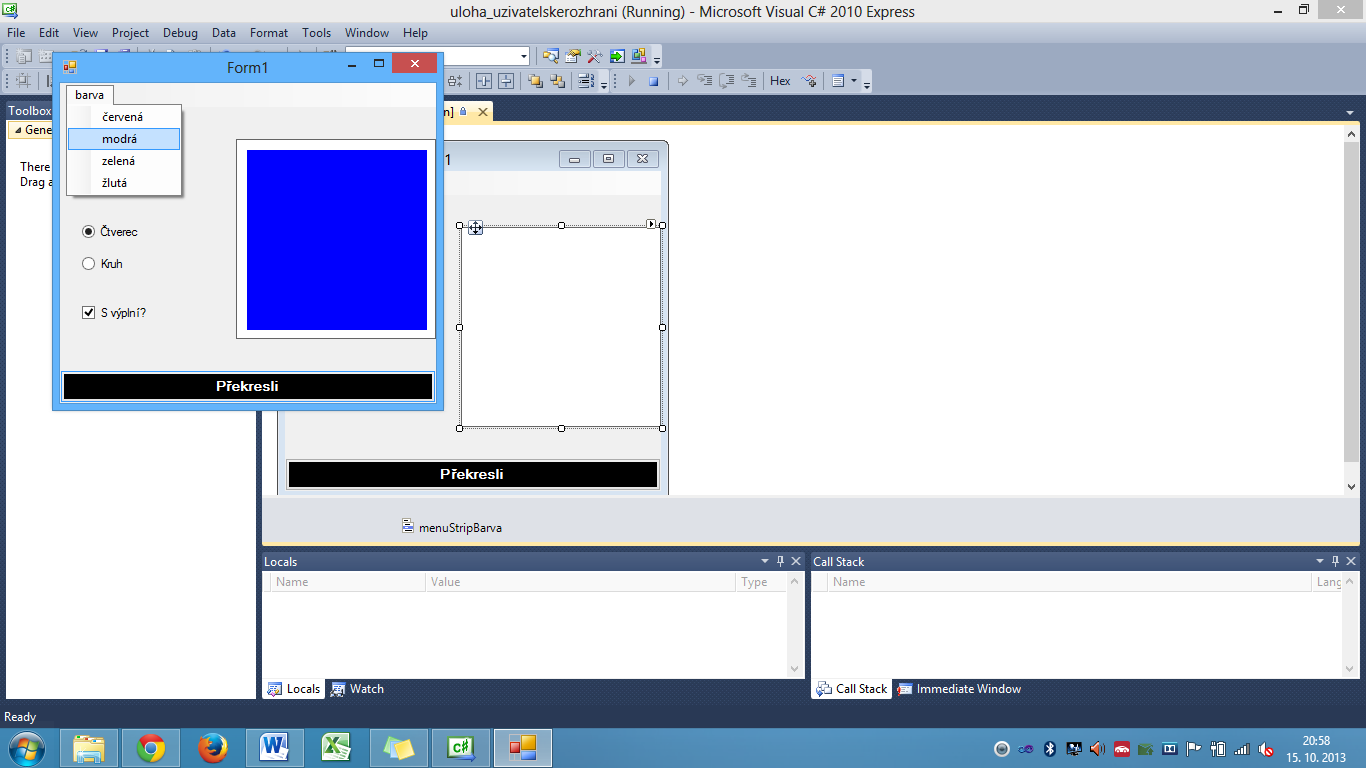
textBoxHexadecimalni.Text = vysledek;

}

}

}

# 6. Uživatelské rozhraní

Vytvořte grafické rozhraní podle vzoru a nakreslete v prostoru prvku Panel dle volby obrazec v příslušné barvě, tvaru, s nebo bez výplně.

K volbě barev použijte ovládací prvek MenuStrip.

namespace uloha\_uzivatelskerozhrani

{

public partial class Form1 : Form

{

Pen mojepero = new Pen(Color.Black);

SolidBrush mujstetec = new SolidBrush(Color.Black);

Graphics kp;

int obrazec;

bool vypln;

public Form1()

{

InitializeComponent();

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

kp = e.Graphics;

mojepero.Width = 10;

if (obrazec == 1)

if (vypln == true)

kp.FillRectangle(mujstetec, 10, 10, 180, 180);

else

kp.DrawRectangle(mojepero, 10, 10, 180, 180);

else

if (vypln == true)

kp.FillEllipse(mujstetec, 10, 10, 180, 180);

else

kp.DrawEllipse(mojepero, 10, 10, 180, 180);

}

private void buttonPrekresli\_Click(object sender, EventArgs e)

{

Refresh();

}

private void červenáToolStripMenuItem\_Click(object sender, EventArgs e)

{

mojepero.Color = Color.Red;

mujstetec.Color = Color.Red;

}

private void modráToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

mojepero.Color = Color.Blue;

mujstetec.Color = Color.Blue;

}

private void zelenáToolStripMenuItem\_Click(object sender, EventArgs e)

{

mojepero.Color = Color.Green;

mujstetec.Color = Color.Green;

}

private void žlutáToolStripMenuItem\_Click(object sender, EventArgs e)

{

mojepero.Color = Color.Yellow;

mujstetec.Color = Color.Yellow;

}

private void radioButtonCtverec\_CheckedChanged(object sender, EventArgs e)

{

obrazec = 1;

}

private void radioButtonKruh\_CheckedChanged(object sender, EventArgs e)

{

obrazec = 2;

}

private void checkBoxVypln\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxVypln.Checked == true)

vypln = true;

else vypln = false;

}

}

}

# 7. Objektově orientované programování

Napište program, který se vždy zeptá, zda chceme losovat další číslo a to se přidá do Listu. Pokud již nebudeme chtít losovat, program vypíše losovaná čísla, seřazená od nejmenšího po největší.

**1) Třída Losovač**

namespace Losovac

{

/// <summary>

/// Losovač k losování a zaznamenávání čísel

/// </summary>

class Losovac

{

/// <summary>

/// Vnitřní kolekce čísel

/// </summary>

private List<int> cisla;

/// <summary>

/// Generátor náhodných čísel

/// </summary>

private Random random;

/// <summary>

/// Vytvoří nový losovač

/// </summary>

public Losovac()

{

random = new Random();

cisla = new List<int>();

}

/// <summary>

/// Vylosuje nové náhodné číslo a to uloží do kolekce

/// </summary>

/// <returns>Nové náhodné číslo</returns>

public int Losuj()

{

int cislo = random.Next(100) + 1;

cisla.Add(cislo);

return cislo;

}

/// <summary>

/// Vrátí string, obsahující výpis seřazených tažených čísel

/// </summary>

/// <returns> výpis seřazených tažených čísel</returns>

public string Vypis()

{

string s = "";

cisla.Sort();

foreach (int i in cisla)

s += i + " ";

return s;

}

}

}

**2) Hlavní program**

namespace Losovac

{

class Program

{

static void Main(string[] args)

{

Losovac losovac = new Losovac();

Console.WriteLine("Vítejte v programu losování.");

char volba = '0';

// hlavní cyklus

while (volba != '3')

{

// výpis možností

Console.WriteLine("1 - Losovat další číslo");

Console.WriteLine("2 - Vypsat čísla");

Console.WriteLine("3 - Konec");

volba = Console.ReadKey().KeyChar;

Console.WriteLine();

// reakce na volbu

switch (volba)

{

case '1':

Console.WriteLine("Padlo číslo: {0}", losovac.Losuj());

break;

case '2':

Console.WriteLine("Padla čísla: {0}", losovac.Vypis());

break;

case '3':

Console.WriteLine("Děkuji za použití programu");

break;

default:

Console.WriteLine("Neplatná volba, zadejte prosím znovu.");

break;

}

}

}

}

}

Napište program, který umožní vytvořit nový objekt *letadlo* s parametry „Rok výroby“, „Typ“ a „Kapacita“. Tyto atributy budou privátní. Třída *Letadlo*, která bude tyto atributy spravovat, bude obsahovat také metodu pro přidávání cestujících a metodu pro výpis objektu *letadlo* jako string. Metoda pro přidávání cestujících ošetří vstup tak, aby se nemohlo stát, že bude překročena kapacita letadla.

namespace UkazkaTvorbyObjektu

{

// Ukázka tvorby objektů - tvoření letadel

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Ukázka tvorby objektů - tvoření letadel");

Console.WriteLine("Zkusíme si použití konstruktoru objektů");

Console.WriteLine("Zadejte datum výroby: (např. 1.1.1990)");

DateTime vyrobeno = DateTime.Parse(Console.ReadLine());

Console.WriteLine("Zadejte název letadla: ");

string nazev = Console.ReadLine();

Console.WriteLine("Zadejte kapacitu: ");

int capacity = int.Parse(Console.ReadLine());

Letadlo letadlo = new Letadlo(vyrobeno, nazev, capacity);

Console.WriteLine("Instance vytvořena, nyní si vyzkoušíme zavolat metodu");

Console.WriteLine("Zadejte počet cestujících: ");

int cestujicich = int.Parse(Console.ReadLine());

letadlo.pridejCestujici(cestujicich);

Console.WriteLine("Podívejme se na výsledek: ");

Console.WriteLine(letadlo);

Console.ReadKey();

}

}

}

namespace UkazkaTvorbyObjektu

{

// deklarace třídy letadlo

class Letadlo

{

// privátní (vnitřní) atributy

private DateTime vyrobeno;

private string jmeno;

private int kapacita;

private int cestujicich = 0;

// konstruktor, umožňující dosadit atributy při vytvoření objektu

public Letadlo(DateTime vyrobeno, string jmeno, int kapacita) {

this.vyrobeno = vyrobeno;

this.jmeno = jmeno;

this.kapacita = kapacita;

}

// metoda přidávající cestující

// proměnnou cestujících jsme nezpřístupnili přímo, ale přes

// tuto metodu proto, aby se nemohlo stát, že bude překročena

// kapacita

public void pridejCestujici(int pocet) {

if ((cestujicich + pocet) <= kapacita)

cestujicich += pocet;

}

// přetížení metody pro výpis objektu jako string

public override string ToString()

{

return jmeno + " " + vyrobeno + " " + cestujicich + "/" + kapacita;

}

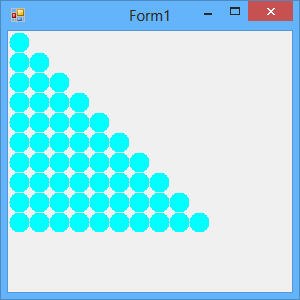
}

}

# 8. Grafika

Nakreslete "trojúhelník" koleček 10x10 o průměru kolečka 20 pixlů.

Možné jsou 4 různé varianty podle umístění rohu s pravým úhlem. Obrazec se vykreslí při startu programu.



namespace uloh\_grafikavcyklu

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

//obrazec se vykreslí při startu programu - jde o událost Paint pro Formulář

Graphics kp = e.Graphics;

//varianta a - pravý úhel v levém horním rohu

/\*

int kam = 10;

int x;

int y=1;

for (int i = 1; i <= 10; i++)

{

x = 1;

for (int j = 1; j <= kam; j++)

{

kp.FillEllipse(Brushes.Aqua, x, y, 20, 20);

x += 20;

}

y += 20;

kam--;

}

\*/

//varianta b - pravý úhel v levém dolním rohu

int kam = 1;

int x;

int y = 1;

for (int i = 1; i <= 10; i++)

{

x = 1;

for (int j = 1; j <= kam; j++)

{

kp.FillEllipse(Brushes.Aqua, x, y, 20, 20);

x += 20;

}

y += 20;

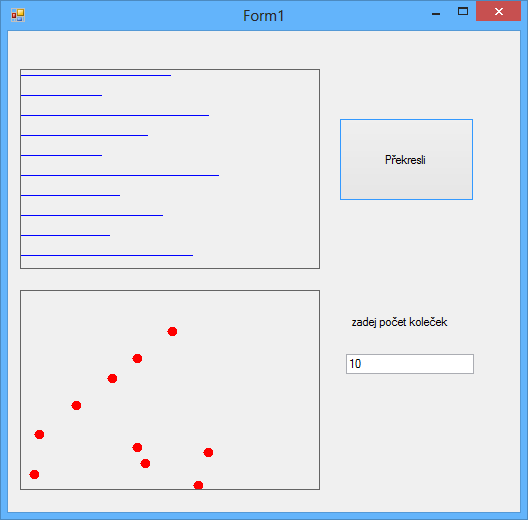
kam++;

}

}

}

}

Př.

a, Nakreslete pod sebou 10 čar o náhodných délkách, maximálně 200 px. Vzdálenost čar 20 px. Zvolte vhodně velikost prvku Panel, který bude mít viditelný rámeček.

b, Nakreslete na náhodných pozicích kolečka o průměru 10 pixlů. Počet koleček zadá uživatel.

namespace uloha\_grafikanahodne

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

Graphics kp = e.Graphics;

Pen mojemodrepero = new Pen(Color.Blue);

Random rnd = new Random();

int souradnicey = 5;

for (int i = 1; i <= 10; i++)

{

int delka = rnd.Next(1, 200);// resp. maximum je panel1.Width

kp.DrawLine(mojemodrepero, 0, souradnicey, delka, souradnicey);

souradnicey += 20;

}

}

private void panel2\_Paint(object sender, PaintEventArgs e)

{

Graphics kp = e.Graphics;

SolidBrush mujcervenystetec = new SolidBrush(Color.Red);

Random rnd = new Random();

string vstup = textBoxPocet.Text;

int pocetkolecek;

//musíme ošetřit stav po spuštění programu, kdy textbox je prázdný - jinak běhová chyba

if (vstup != "")

{

pocetkolecek = Convert.ToInt32(vstup);

int pozicex;

int pozicey;

for (int i = 1; i <= pocetkolecek; i++)

{

pozicex = rnd.Next(1, 190); //resp. maximum je panel1.Width

pozicey = rnd.Next(1, 190); //maximum je panel1.Height

kp.FillEllipse(mujcervenystetec, pozicex, pozicey, 10, 10);

}

}

}

private void tlacitko1\_Click(object sender, EventArgs e)

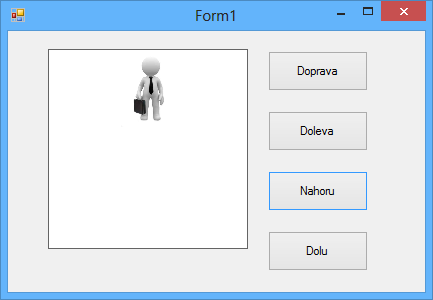
{

Refresh();

}

}

}



Př.

Připravte si grafické rozhraní s panelem o rozměrech 200 x 200, rámečkem a pictureboxem s vloženým obrázkem.

Upravte barvu pozadí na bílou a velikost boxu podle proporcí obrázku. Chceme, aby se panáček pohyboval podle tlačítek a to vždy pouze jedním směrem. Pohyb bude ukončen „nárazem do stěny“ nebo stiskem jiného tlačítka.

Optimalizujte rychlost pohybu (zpomalte) pomocí prvku Timer s intervalem cca 10.

namespace uloha\_grafikasobrazkem

{

public partial class Form1 : Form

{

//proměnné indikující, o jaký směr se jedná

//h=1 doprava, h=2 doleva v=1 dolů v=2 nahoru

int h;

int v;

public Form1()

{

InitializeComponent();

}

//picturebox - vložíme obrázek, upravíme vlastnost Sizemode na Zoom

//4 tlačítka - doprava, doleva, nahoru, dolů - vždy do kraje panelu

//protože je pohyb velmi rychlý, je nutné použít timer a upravit jeho interval na cca 10 tj. vlastní příkazy budou v metodě timeru

private void buttonDoprava\_Click(object sender, EventArgs e)

{

h = 1;

v = 0;

//horizontální pohyb je doprava, vertikální je zablokovaný, spouštíme Timer

timer1.Start();

}

private void buttonDoleva\_Click(object sender, EventArgs e)

{

h = 2;

v = 0;

//horizontální pohyb je doleva, vertikální je zablokovaný

timer1.Start();

}

private void buttonDolu\_Click(object sender, EventArgs e)

{

v = 1;

h = 0;

//vertikální pohyb je dolů horizontální je zablokovaný

timer1.Start();

}

private void buttonNahoru\_Click(object sender, EventArgs e)

{

v = 2;

h = 0;

//vertikální pohyb je nahoru, horizontální je zablokovaný

timer1.Start();

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (h == 1)

{

//vlastnost left určuje pozici levého okraje pictureboxu, porovnáváme s šířkou prvků

if (pictureBoxPanacek.Left < panel1.Width - pictureBoxPanacek.Width)

pictureBoxPanacek.Left += 1;

else

//ukončujeme běh timeru

timer1.Stop();

}

if (h == 2)

{

if (pictureBoxPanacek.Left > 0)

pictureBoxPanacek.Left -= 1;

else

timer1.Stop();

}

if (v == 1)

{

//vlastnost top určuje pozici horního okraje pictureboxu, porovnáváme s výškou prvků

if (pictureBoxPanacek.Top < panel1.Height - pictureBoxPanacek.Height)

pictureBoxPanacek.Top += 1;

else

timer1.Stop();

}

if (v == 2)

{

if (pictureBoxPanacek.Top > 0)

pictureBoxPanacek.Top -= 1;

else

timer1.Stop();

}

}

}

}