| | |
|---|---|
| Číslo a název šablony | III/2 Inovace a zkvalitnění výuky prostřednictvím ICT |
| Číslo didaktického materiálu | EU-OPVK-VT-III/2-ŠR-303 |
| Druh didaktického materiálu | DUM |
| Autor | RNDr. Václava Šrůtková |
| Jazyk | čeština |
| Téma sady didaktických materiálů | Programování v C# v příkladech III |
| Téma didaktického materiálu | Indexované seznamy |
| Vyučovací předmět | Seminář z informatiky |
| Cílová skupina (ročník) | Žáci ve věku 17–18 let |
| Úroveň žáků | Středně pokročilí |
| Časový rozsah | 1–2 vyučovací hodiny |
| Klíčová slova | Indexové seznamy, rozvírací seznam |
| Anotace | Studenti se učí používat rozvírací seznam – komponentu listBox, programovat vlastní seznamy a pracovat s nimi |
| Použité zdroje | DRÓZD, Januš a Rudolf KRYL. *Začínáme s programováním*. 1.vyd. Praha: Grada, 1992, 306 s. ISBN 80-854-2441-X.<br><br>ELLER, Frank. *C# - začínáme programovat: podrobný průvodce začínajícího uživatele*. 1. vyd. Praha: Grada, 2002, 240 s. ISBN 80-247-0324-6.<br><br>LIBICHER, Ivan a Pavel TÖPFER. *Od problému k algoritmu a programu: sbírka řešených úloh z programování*. 1. vyd. Praha: Grada, 1992, 119 s. Educa '99. ISBN 80-854-2482-7.<br><br>TÖPFER, Pavel. *Algoritmy a programovací techniky*. 1. vyd. Praha: Prometheus, 1995, 299 s. ISBN 80-858-4983-6.<br><br>TÖPFEROVÁ, Dana a Pavel TÖPFER. *Sbírka úloh z programování*. Vyd. 1. Praha: Grada, 1992, 98 s. Educa '99. ISBN 80-854-2499-1.<br><br>VYSTAVĚL, Radek. *Moderní programování: učebnice pro pokročilé*. Ondřejov: moderníProgramování, 2011, 149 s. ISBN 978-80-903951-7-6.<br><br>VYSTAVĚL, Radek. *Moderní programování: učebnice pro středně pokročilé*. Ondřejov: moderníProgramování s.r.o, 2008. ISBN 978-80-903951-2-1. |
| Typy k metodickému postupu učitele, doporučené výukové metody, způsob hodnocení, typy k individualizované výuce apod. | Text je možno využít ke společné práci, samostatné přípravě studentů, domácímu studiu apod.<br>Při společné práci je vhodné nejprve obtížnější úlohy rozebrat, potom společně se studenty implementovat na počítači. (Rozbor nejlépe na tabuli, synchronní řešení s promítáním)<br>Prezentace obsahuje stručné shrnutí poznatků |

| | |
|---|---|
| | potřebných pro řešení příkladů. V pracovním listu je zadání cvičení – většinou se jedná o úlohy, které by měli studenti naprogramovat samostatně. Není nutné, aby všichni zpracovali všechno, vhodné je diferencovat podle jejich zájmu a schopností. Obtížnější úlohy jsou označeny hvězdičkou. Součástí materiálu je zdrojový kód těchto příkladů.<br>Návrh způsobu hodnocení:<br>ohodnocení samostatné práce během hodiny např. podle volby a počtu úloh a elaborace řešení (efektivnost, komentáře…). |

**Metodický list k didaktickému materiálu**

**Prohlášení autora**

Tento materiál je originálním autorským dílem. K vytvoření tohoto didaktického materiálu nebyly použity žádné externí zdroje s výjimkou zdrojů citovaných v metodickém listu.

Obrázky (schémata a snímky obrazovek) pocházejí od autora.

# 303. Indexované seznamy

Podobně jako pole se používají k uložení více hodnot, které jsou přístupné pomocí svých indexů, ale na rozdíl od pole mohou dynamicky růst. (Jejich délka není omezena deklarací)

Jsou to instance třídy **List<Typ>**, kde se v lomených závorkách uvádí skutečný typ položek seznamu. Při standardním vytvoření bezparametrickým konstruktorem
`List<int> cisla = new List<int>();` je seznam na začátku prázdný, položky se do něj přidávají metodou **Add**. Aktuální počet prvků seznamu udává vlastnost **Count** (U polí **Length**).

**Příklad 1.**

Naplníme seznam náhodnými čísly, zobrazíme ho, přidáme číslo zadané uživatelem na konec a opět zobrazíme.

Ke zobrazení bychom mohli použít víceřádkové textové pole, ale přímo pro seznamy je v C# komponenta **listBox**. (Vlastně rozvírací seznam) Seznam jeho položek je pak dostupný jako **ListBox.Items**.

Na formulář umístíme dva – listBoxVstup a listBoxVystup.

```
namespace _38_Seznam

{

    public partial class Form1 : Form
```

```
{
    List<int> cisla = new List<int>();
    Random nahoda = new Random();
    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        //naplnění seznamu 10 náhodnými čísly z generátoru
        for (int i = 0; i < 10; i++)
        {
            int x = nahoda.Next(100);
            cisla.Add(x);
        }
    }

    private void buttonUkaz_Click(object sender, EventArgs e)
    {
      //zobrazení seznamu do listBoxu
        for (int i = 0; i < cisla.Count; i++)
                {
                listBox.Items.Add(cisla[i]);
                }
    }
```

Všimněte si, že při přidávání do listBoxu není nutné konvertovat číslo na text.

**Příklad 2.**

Se seznamem budeme provádět další operace a pak ho zobrazovat – co kdybychom si k tomu účelu vyrobili metodu? Pak bychom mohli program upravit následujícím způsobem. Prvním parametrem metody je zobrazovaný seznam, druhým listBox, do kterého budeme zobrazovat, abychom mohli

porovnávat původní a pozměněný seznam. Typ metody je void – slouží ke zobrazování, nemá návratovou hodnotu.

```
  private void Vypis(List<int> x,ListBox LB)
        {

           //zobrazení seznamu x do listBoxu LB
           for (int i = 0; i < x.Count; i++)
            {
                LB.Items.Add(x[i]);
            }

        }
private void buttonUkaz_Click(object sender, EventArgs e)

         {

             Vypis(cisla,listBox);

         }
```

**Operace nad seznamem**

- Naplnění seznamu
- Zobrazení konkrétního prvku
- Vložení prvku na určité místo
- Odstranění zadaného prvku

**Naplnění seznamu**

Můžeme realizovat postupnou metodou **Add(prvek)** z libovolného zdroje. (Vstup od uživatele, vstupní soubor, náhodná čísla…) – viz předchozí příklad.

**Zobrazení konkrétního prvku**

K jednotlivým prvkům přistupujeme pomocí indexu, pokud je seznam zobrazen v listBoxu a uživatel vybere některý prvek, jeho index udává vlastnost **SelectedIndex**. Pokud není vybrán žádný prvek, je hodnota této vlastnosti rovna – 1.
Pokud chceme v listBoxu zrušit výběr, můžeme to provést příkazem:

```
listBoxVstup.SelectedIndex = -1;
```

Vymazání celého listBoxu: `listBoxVystup.Items.Clear();`

**Vložení prvku na určité místo**

Metoda **Insert(index, prvek) –** 1. parametr je pozice, na kterou se bude vkládat, 2. parametr vkládaný prvek.

**Odstranění zadaného prvku**

Metoda **RemovaAt(index),** parametr je opět index.

Stejné metody lze použít pro listBox, původní seznam se tím ovšem nezmění.
(Např `listBoxVstup.Items.RemoveAt(3); odstraní 4. Prvek`)

Všechno si vyzkoušíme v následujícím programu.

```
private void buttonSelInd_Click(object sender, EventArgs e)

        {   //zobrazí index vybrané položky i vybranou položku.

            int i = listBox.SelectedIndex;

            MessageBox.Show(i.ToString() + ". položka: " +
listBox.Items[i].ToString());

        }

         private void buttonVloz_Click(object sender, EventArgs e)
        {
            //vloží nové číslo do seznamu před vybraný prvek
            int i = listBoxVstup.SelectedIndex;
            if (i >= 0) //prvek se v seznamu nachází
                cisla.Insert(i, Convert.ToInt32(textBoxVklad.Text));
            else
                cisla.Add(Convert.ToInt32(textBoxVklad.Text));
                //vložení na konec
            textBoxVklad.Text = null;
            textBoxVklad.Focus();
            //Původní seznam čísel se změnil
            Vypis(cisla, listBoxVystup);
        }


private void buttonSmaz_Click(object sender, EventArgs e)
        {
            int i = listBoxVstup.SelectedIndex;
            cisla.RemoveAt(i);
        }
```

Poznámka: pokud byste chtěli kopírovat seznam prostým přiřazením, zkopíruje se pouze odkaz a obě proměnné budou ukazovat na tentýž seznam. Chceme-li původní seznam měnit, je třeba vytvořit nový seznam na základě existujícího:

List<Typ> seznamNovy = **new** List<Typ>(seznamPůvodní)

## Pracovní list

### Cvičení

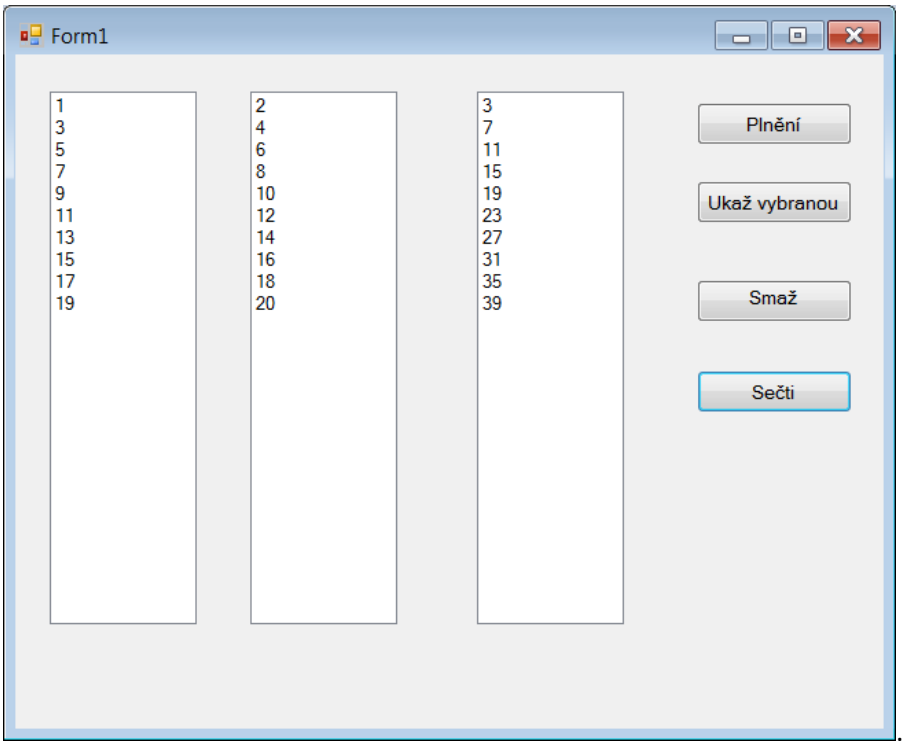
1. Procvičte si práci se dvěma listBoxy Lichý a Sudý. (Nebudete vytvářet seznamy, ale pracovat pouze s obsahem listBoxů.)

a) Na stisknutí tlačítka naplníte Lichý prvními 10 lichými a Sudý prvními deseti sudými čísly.

b) Vyzkoušejte si operace odstranění vybraného čísla, zobrazení vybraného čísla

c) Zobrazte zprávu o tom, je-li uživatelem zadané číslo v některém listBoxu.

d) (*) Sečtěte čísla ve stejných řádcích a zobrazte je do třetího listBoxu Součet, pokud mají seznamy stejný počet položek.




2. Přidejte na okno tlačítko, jehož stisknutím se nová položka vloží za vybranou položku.

## Řešení

1.

```
private void buttonPln_Click(object sender, EventArgs e)
        {
            //plnění rozvíracích seznamů čísly
            for (int i = 1; i <= 10; i++)
            {
                listBoxLicha.Items.Add(2 * i - 1);
                listBoxSuda.Items.Add(2 * i);

            }

        }


        private void buttonUkaz_Click(object sender, EventArgs e)
        {
            //Zobrazení zprávy o vybraných položkách
            int i = listBoxLicha.SelectedIndex;
            if (i >= 0)
```

```
                MessageBox.Show(listBoxLicha.Items[i].ToString());
            int j = listBoxSuda.SelectedIndex;
            if (j >= 0)
                MessageBox.Show(listBoxSuda.Items[j].ToString());

        }

        private void buttonSmaz_Click(object sender, EventArgs e)
        {
            //odstranění vybraných položek
            int i = listBoxLicha.SelectedIndex;
            if (i >= 0)
                listBoxLicha.Items.RemoveAt(i);
            int j = listBoxSuda.SelectedIndex;
            if (j >= 0)
                listBoxSuda.Items.RemoveAt(j);
        }

        private void buttonSecti_Click(object sender, EventArgs e)
        {
            //naplnění součtového seznamu
            if (listBoxSuda.Items.Count == listBoxSuda.Items.Count)
                for (int i = 0; i < 10; i++)
                {
                    int s = Convert.ToInt32(listBoxLicha.Items[i]) +
Convert.ToInt32(listBoxSuda.Items[i]);
                    listBoxSoucet.Items.Add(s);
                }
            else
                MessageBox.Show("Pro různé počty nelze sečíst!");

        }
    }


2
private void button1_Click(object sender, EventArgs e)
        {
            //pokud je vybraná položka poslední, přidáme ji metodou Add.
            int i = listBoxVstup.SelectedIndex;
            if (i == cisla.Count)
                cisla.Add(Convert.ToInt32(textBoxVklad.Text));
            else
            {
                if (i >= 0) //něco je vybráno
                {
                    cisla.Insert(i+1, Convert.ToInt32(textBoxVklad.Text));
                    textBoxVklad.Text = null;
                    Vypis(cisla, listBoxVystup);
                }
            }
        }
```

## Důležité

Seznam může dynamicky růst. (délka není omezena deklarací)

Instance třídy **List<Typ>**, kde se v lomených závorkách uvádí skutečný typ položek seznamu.

Při standardním vytvoření bezparametrickým konstruktorem `List<int> cisla = new List<int>();`

**Vlastnosti a metody**:

**Count** – počet prvků

**Add**(prvek) – přidání na konec

**Insert**(index, prvek) – přidání na aktuální pozici

**RemoveAt**(index) – odstranění prvku z aktuální pozice

**ListBox** – komponenta na vkládání seznamů libovolného typu

**SelectedIndex** – index vybrané položky

**Items** – seznam