Jazyk C# a platforma .NET

Katedra softwarového inženýrství
Fakulta informačních technologií
České vysoké učení technické v Praze

Syntaxe jazyka C# - 1. část
BI-DNP

```
// Zaklady jazyka C# (prvni cast)

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 JazykCS {
    public partial class Form1 : Form {
        public Form1() {
            InitializeComponent();
        }

        private void btnPromenne_Click(object sender, EventArgs e) {
            // celociselne typy
            int i, j, k;
            short s1, s2, s3;

            // tabulka datovych typu jazyka C# - viz help

            i = 1;
            s1 = 2;
            j = s1;

            // s1 = i;          // nelze 32 bit do 16
            s1 = (short)i;      // provede - pretypovani (casting) NUTNE!!

            s1 = 1; s2 = 2;
            // s3 = s1 + s2;    // nelze - MINIMALNI celosiselny typ je int!!

            /* prelozi takto:
               int t1, t2;
               t1 = s1;
               t2 = s2;
               s3 = t1 + t2;
            */
            s3 = (short)(s1 + s2);  // OK

            // pohybliva desetinna carka
            double d = 123.45e3;

            // decimal - uklada a zpracovava v desitkove soustave
            // (nikoli v modifikovane dvojkove)
            decimal dec = 123.45m;  // suffix m = ulozit literal ve tvaru
                                    // decimal (ne v implicitnim double)

            float f = 3.1415f;  // suffix f – ulozi jako float, nikoli double
```

```
        // boolean
        bool b;
        b = true;
        b = i > 0;

        // b = 1;   // nelze - 1 neni typ boolean

        // object - lze pouzit pro ulozeni ruznych typu
        object o;
        o = 1;
        o = "Ahoj";
        o = 1.2;

        // znaky
        char znak;
        znak = 'A';

        // retezec
        string retezec = "ABCDEF";

        // escape sekvence: \t - tabelator, \r - carriage return,
        // \n - line feed, \\ - backslash, \" - uvozovka, \' - apostrof...
        retezec = "Prvni\r\nDruhy\r\nTreti";
        txtVystup.Text = retezec;

        // Environment.NewLine  // prenositelna nova radka

        txtVystup.Text = "C:\\Data\\Texty\\Dopisy\\Dopis.txt";

        // @ pred retezcem vypne \ jako zacatek escape sekvence
        txtVystup.Text = @"C:\Data\Texty\Dopisy\Dopis.txt";

        // string jako pole (presneji indexer)
        retezec = "ABCD";
        znak = retezec[2];      // treti znak - C
        txtVystup.Text = znak.ToString();

        // retezec[2] = 'x';    // nelze - string jako pole je Read Only!!

        // definice konstant
        const int CISLO = 5;

        // CISLO = 1;   // nelze - konstanta

        const int SEC_MIN = 60;
        const int MIN_HOD = 60;
        const int HOD_DEN = 24;

        // konstantni vyrazy (museji byt vyhodnotitelne v dobe prekladu)
        const int SEC_DEN = SEC_MIN * MIN_HOD * HOD_DEN;

    }
```

```
private void btnOperatory_Click(object sender, EventArgs e) {
    int i, j, k;

    // operatory aritmeticke + - * / % (modulo - zbytek po deleni)
    // celociselne deleni - jsou-li oba operandy celociselne

    i = int.MaxValue;
    txtVystup.Text = i.ToString();

    // celociselne preteceni (exception) je v C# standardne vypnute
    checked {   // v tomto bloku testovat preteceni (unchecked - ne)
        // i = i + 1;  // nepretece
    }
    txtVystup.Text = i.ToString();

    // v pohyblive radove carce nepretece NIKDY (nevyvola vyjimku)!

    double d = double.MaxValue;
    d = d + d;
    if (double.IsInfinity(d)) // test preteceni v pohyblive radove
                              // carce
        txtVystup.Text = "!!! NEKONECNO !!!";
    else
        txtVystup.Text = d.ToString();

    // operator retezeni +
    txtVystup.Text = "ABCD" + "xyz";

    // automaticka konverze při retezeni
    txtVystup.Text = "Cislo = " + 5;    // + mezi retezcem a
                // NEretezcem - na NEretezec pouzije C# .ToString()

    // pozor na poradi pri retezeni!! (Mozno zmenit zavorkami)
    txtVystup.Text = "Soucet je " + 2 + 3;    // vypise Soucet je 23

    txtVystup.Text = 2 + 3 + " je soucet";    // vypise 5 je soucet

    // operator prirazeni = (vraci to, co se prirazuje);
    // asociativni zprava
    i = j = k = 5;          // priradi 5 do vsech promennych
    i = (j = (k = 5));      // zpracuje takto

    // operatory relacni < <= > >= != (nerovno) == (rovno - pozor!)
    /* // chyba v jazyce C
       i = 5;
       if (i = 1) {    // mel napsat ==
           // provede vzdy!!
       }
    */
```

```
        // stejna chyba v jazyce C#
        i = -2;
        k = 3;
        bool b = i > 0;
        if (b = (k == 3)) { // chtel napsat ==
            // provede vzdy
        }

        // operatory logicke && - And, || - Or, ! - Not
        if ((i > 0) && (k < 5)) { // napr.
            // ...
        }

        // operatory bitove-logicke & - bitove And, | - bitove Or,
        // ^ - bitove Xor, ~ bitove Not
        i = 3;      // ... 0011
        k = 5;      // ... 0101
        j = k & i;  // ... 0001

        // operatory bitovych posuvu: vlevo << a vpravo >>
        i = 3;      // ... 0011
        k = i << 2; // .00 1100
        // pri posunu vpravo provadi LOGICKY posun (doplnuje nuly) pro
        // NEznamenkove typy, ARITMETICKY (opakuje znamenko) pro
        // znamenkove typy

        // bitove-logicke operatory lze pouzit jako LOGICKE:
        if ((i > 0) & (k == 3)){
            // ...
        }

        // logicke operatory se vyhodnocuji ZKRACENE (pouze, co je treba)!!
        i = -2;
        if ((i > 0) && (1 < k++)){  // k++ se NEPROVEDE!!
            // ...
        }

        // bitove-logicke operatory se vyhodnocuji NEzkracene
        i = -2;
        if ((i > 0) & (1 < k++)){  // k++ se PROVEDE!!
            // ...
        }

        // ternalni (podmineny) operator
        k = (i > j) ? i : j;

        // neprehanet!!
        k += (i++ > --i) ? ++i + i-- : i-- - ++i;

    }
```

```
int z;  // definice NElokalni promenne (neni definovana ve funkci)

private void btnPrikazy_Click(object sender, EventArgs e) {
    int i, j, k;

    // prirazovaci
    i = 5;

    // podminene
    if (i > 0) j = 3;

    if (i > 0) {
        i = 1;
        j = 2;
        k = 3;
    }
    else {
        k = 3;
        j = 2;
    }

    // cykly
    i = 8;
    while (i > 0) {
        // ...
        i--;
        // ...
        if (i == 4) break;  // vyskoci z cyklu
        // ...
        if ((i % 10) == 0) continue; // zacne dalsi cyklus
        // prikazy preskocene pomoci continue
    }

    // break a continue i v dalších typech cyklu

    i = 8;
    do {
        i--;
    }
    while (i > 0);

    for (i = 0; i < 8; i++) {
        // ...
    }

    for (i = 0; i < 8; i += 3) { // s krokem 3
        // ...
    }

    for (i = 7; i >= 0; i--) {   // sestupny
        // ...
    }
```

```
// for (; ; ) { }   // nekonecny cyklus

for (int n = 0; n < 8; n++) {   // definovani parametru cyklu
                                // primo ve for
    // ...
}

// k = n;                       // nelze - n je LOKALNI v cyklu

for (i = 0, j = 9; i < j; i++, j--) {   // operator carka
    // ...
}

/*
foreach (typPrvku prvek in kolekcePrvku) {
    // ...
}
*/

// vetveni (switch)
char roman = 'x';
int arab;

switch (roman) {
    case 'i':
        arab = 1;
        break;
    case 'v':
        arab = 5;
        break;
    case 'x':
        arab = 10;
        break;
    case 'l':
        arab = 50;
        break;
    // ...
    default:
        arab = 0;
        break;
}

txtVystup.Text = arab.ToString();
```

```
// propadani s prazdnymi case je mozne
char znak = 'i';
bool jeSamohlaska;

switch (znak) {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
    case 'y':
        jeSamohlaska = true;
        break;
    default:
        jeSamohlaska = false;
        break;
}

// propadani u neprazdnych case je zakazane (MUSI by break,
// return, goto, ...) Hloupy, ale funkcni priklad:
int cislice = 8;
int do10 = 0;

switch (cislice) {
    case 0:
        do10++;
        // u NEPRAZDNYCH case je "propadani"nutne prikazat!!
        goto case 1;
    case 1:
        do10++;
        goto case 2;
    case 2:
        do10++;
        goto case 3;
    case 3:
        do10++;
        goto case 4;
    case 4:
        do10++;
        goto case 5;
    case 5:
        do10++;
        goto case 6;
    case 6:
        do10++;
        goto case 7;
    case 7:
        do10++;
        goto case 8;
    case 8:
        do10++;
        goto case 9;
```

```
                case 9:
                    do10++;
                    break;
            }


            // prikaz skoku
            if (i > 0) goto Hop;

            // ...

            Hop: ;    // navesti


            // LOKALNI promenne VNE a UVNITR bloku NESMEJI mit stejne jmena

            // int x;   // nelze
            {
                int x;
                x = 1;

                int z;  // NElokalni a LOKALNI promenne MOHOU mit stejne
                        // jmeno!!
                z = 3;

            Vnitrni: ;

            }

            // goto Vnitrni;   // nelze skocit dovnitr bloku

            // x = 3;          // nelze - x je lokalni v bloku

            { // zacatek bloku
                int x;

            } // konec bloku
        }
    }
}
```