



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání



Základy programování a algoritmizace

Booleanovské výrazy

Erik Král



2020

Informace o autorech:

Ing. et Ing. Erik Král, Ph.D.

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Nad Stráněmi 4511

760 05 Zlín

ekral@utb.cz



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání



OBSAH

OBSAH.....	3
1 ÚVOD.....	4
1.1 BOOLEANOVSKÉ VÝRAZY.....	4
1.2 ŘEŠENÉ PŘÍKLADY	7
SEZNAM POUŽITÉ LITERATURY	13



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání


MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



1 ÚVOD

V tomto materiálu se seznámíme s booleanovskými výrazy, tedy výrazy, které vracejí typ *boolean* [1] a projdeme si i příklady na procvičení.

1.1 Booleanovské výrazy

Booleovský výraz vrací jako výslednou hodnotu typ *bool*. Nejčastěji jej používáme v podmíněných příkazech výběru (*selection statements*) a příkazech pro tvorbu cyklu (*iteration statements*). Příklady booleanovských výrazů jsou relační a logické operátory.

Relační operátory představují následující operátory:

```
int x = 2;
int y = 3;
bool vysledek;

vysledek = x == y; // rovnost
vysledek = x != y; // nerovnost
vysledek = x < y;  // mensi nez
vysledek = x > y;  // vetsi nez
vysledek = x <= y; // nensi nebo rovno
vysledek = x >= y; // vetsi nebo rovno
```

Rovnost tedy zapisujeme následujícím způsobem, jeden znak by totiž představoval operátor přiřazení.

```
x == y
```

A nerovnost zapisujeme takto:

```
x != y
```



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MŠMT
MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



Dalšími booleanovskými operátory jsou logické operátory.

Logický AND se zapisuje pomocí znaků `&&`. Výsledkem operace `x && y` je *true*, pokud *x* a zároveň *y* jsou *true*. Jinak je výsledkem *false*. V následujícím příkladu bude mít proměnná *ok* hodnotu *true*, pokud je *x > y* a zároveň je *x > 0*.

```
int x = 3;
int y = 5;
bool ok = (x > y) && (x > 0);
```

Logický OR se zapisuje pomocí znaků `||`. Výsledkem operace `x || y` je *true*, pokud *x* a nebo *y* jsou *true*. Jinak je výsledkem *false*.

V následujícím příkladu bude mít proměnná *ok* hodnotu *true*, pokud je *x > y* a nebo je *x > 0*.

```
int x = 3;
int y = 5;
bool ok = (x > y) || (x > 0);
```

Posledním logickým operátorem je logická negace operandu. Pokud je operand *true*, výsledkem je *false*. Pokud je operand *false*, výsledkem je *true*. V následujícím příkladu bude mít proměnná *notOK* hodnotu *false*.

```
bool ok = true;
bool notOK = !ok;
```





Pro zopakování uvedme tři další příklady s komentářem, nejprve opět definujme proměnné *x*, *y* a *vysledek*:

```
int x = 2;  
int y = 3;  
bool vysledek;
```

výsledek bude pravda pokud je *x* menší než *y* a zároveň *y* je rovno 3:

```
vysledek = (x < y) && (y == 3);
```

výsledek bude pravda pokud je *x* menší než *y* a nebo *y* je rovno 3:

```
vysledek = (x < y) || (y == 2);
```

operátor *!* neguje výsledek předchozí operace, výraz je pravda, pokud je *x* větší nebo rovno *y*:

```
vysledek = !(x < y);
```



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MŠMT
MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



1.2 Řešené příklady

Nyní si ukážeme příklady s kompletním kódem a případně okomentujeme jednotlivá řešení.

První příklad shrnuje vše, co jsme zatím probrali.

```
using System;

namespace MujDruhyProjekt
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 2;
            int y = 3;

            bool vysledek;

            vysledek = x == y; // rovnost
            vysledek = x != y; // nerovnost
            vysledek = x < y;   // mensi nez
            vysledek = x > y;   // vetsi nez
            vysledek = x <= y;  // nensi nebo rovno
            vysledek = x >= y;  // vetsi nebo rovno

            vysledek = (x < y) && (y == 3);
            vysledek = (x < y) || (y == 2);
            vysledek = !(x < y);

            Console.WriteLine("Hello World!");
        }
    }
}
```





Druhý příklad určí, zda je trojúhelník pravoúhlý s pomocí pythagorovy věty a zda trojúhelník existuje dle trojúhelníkové nerovnosti [2].

```
using System;

namespace MujDruhyProjekt
{
    class Program
    {
        static void Main(string[] args)
        {
            double a = 3.0;
            double b = 4.0;
            double c = 5.0;

            bool vysledek;

            vysledek = a * a + b * b == c * c;

            Console.WriteLine($"Je pravouhly (True ano, False ne): {vysledek}");

            vysledek = (a + b > c) && (a + c > b) && (b + c > a);

            Console.WriteLine($"Existuje (True ano, False ne): {vysledek}");
        }
    }
}
```

Konkrétně následující řádek určuje, zda je trojúhelník pravoúhlý. Je potřeba počítat s možnou chybou zaokrouhlování a v reálném programu bychom měli počítat s určitou odchylkou.

```
vysledek = a * a + b * b == c * c;
```

A následující řádek testuje trojúhelníkovou nerovnost.

```
vysledek = (a + b > c) && (a + c > b) && (b + c > a);
```



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání



MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



Ve třetím příkladu máme dva výsledky testů studentů, *t1* a *t2* a testujeme různé hypotézy, které níže postupně popíšeme.

Nejdříve níže uvedeme kompletní kód.

```
using System;

namespace MujDruhyProjekt
{
    class Program
    {
        static void Main(string[] args)
        {
            double t1 = 40.0;
            double t2 = 70.0;

            Console.WriteLine($"test1: {t1} test2: {t2}");

            bool vysledek;

            vysledek = t1 > 45.0 || t2 > 45.0;
            Console.WriteLine($"Splnil alespon jeden test: {vysledek}");

            vysledek = t1 <= 45.0 && t2 <= 45.0;
            Console.WriteLine($"Nesplnil zadny test: {vysledek}");

            vysledek = t1 > 45.0 && t2 > 45.0;
            Console.WriteLine($"Splnil oba testy: {vysledek}");

            vysledek = !(t1 <= 45.0 && t2 <= 45.0);
            Console.WriteLine($"Neplati ze nesplnil oba testy: {vysledek}");
        }
    }
}
```





První výraz vrátí *true*, pokud student splnil alespoň jeden ze dvou testů za více než 45 bodů:

```
vysledek = t1 > 45.0 || t2 > 45.0;
```

druhý výraz vrátí *true*, pokud student nesplnil žádný ze dvou testů za více než 45 bodů:

```
vysledek = t1 <= 45.0 && t2 <= 45.0;
```

třetí výraz vrátí *true*, pokud student splnil každý ze dvou testů za více než 45 bodů:

```
vysledek = t1 > 45.0 && t2 > 45.0;
```

a poslední výraz vrátí *true*, pokud je alespoň jeden ze dvou testů splněný za více než 45 bodů, ale tentokrát využívá negaci. Konkrétně nejprve otestujeme, zda jsou oba testy rovny nebo menší 45 bodů a tento výraz potom znegujeme.

```
vysledek = !(t1 <= 45.0 && t2 <= 45.0);
```





Poslední příklad je podobný na předcházející, ale tentokrát psali studenti tři testy.

```
using System;

namespace MujDruhyProjekt
{
    class Program
    {
        static void Main(string[] args)
        {
            double t1 = 30.0;
            double t2 = 40.0;
            double t3 = 70.0;

            Console.WriteLine($"test1: {t1}, test2: {t2}, test3: {t3}");
            bool vysledek;

            vysledek = (t1 > 45.0) || (t2 > 45.0) || (t3 > 45.0);

            Console.WriteLine($"Splnil alespon jeden test ze tri: {vysledek}");

            vysledek = (t1 > 45.0) && (t2 > 45.0) && (t3 > 45.0);

            Console.WriteLine($"Splnil vsechny tri testy: {vysledek}");

            vysledek = ((t1 > 45.0) && (t2 > 45.0))
                || ((t1 > 45.0) && (t3 > 45.0))
                || ((t2 > 45.0) && (t3 > 45.0));

            Console.WriteLine($"Splnil alespon dva testy ze tri: {vysledek}");
        }
    }
}
```





První výraz vrátí *true*, pokud student splnil alespoň jeden ze tří testů za více než 45 bodů:

```
vysledek = (t1 > 45.0) || (t2 > 45.0) || (t3 > 45.0);
```

druhý výraz vrátí *true*, pokud student splnil všechny tři testy, každý za více než 45 bodů:

```
vysledek = (t1 > 45.0) && (t2 > 45.0) && (t3 > 45.0);
```

poslední výraz vrátí *true*, pokud student splnil alespoň dva ze tří testů za více než 45 bodů, testujeme tedy všechny tři varianty:

```
vysledek = ((t1 > 45.0) && (t2 > 45.0))  
           || ((t1 > 45.0) && (t3 > 45.0))  
           || ((t2 > 45.0) && (t3 > 45.0));
```





SEZNAM POUŽITÉ LITERATURY

- [1] Boolean logical operators - C# reference | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 03.01.2021]. Dostupné z: <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/boolean-logical-operators>
- [2] Trojúhelník – Matematika.cz [online]. Copyright © 2006 [cit. 02.10.2020]. Dostupné z: <https://matematika.cz/popis-trojuhelniku>



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

