



# Základy programování a algoritmizace Zásobník

# 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



# **OBSAH**

<b>OBSAH</b>		
	VOD	
	Opakování pojmů	
	ZÁSOBNÍK (STACK)	
1.3	ALGORITMUS VÝPOČTU KOŘENŮ KVADRATICKÉ ROVNICE	
1.4	POPIS PROGRAMU Z HLEDISKA ZÁSOBNÍKU	
1.5	ZÁSOBNÍK A PARAMETRY METODY	16
SEZNA	M POUŽITÉ LITERATURY	30





# 1 ÚVOD

V tomto materiálu se seznámíme s tím, jakým způsobem program alokuje automaticky paměť pro lokální proměnné na zásobníku.

### 1.1 Opakování pojmů

Nejdříve se zopakujme základní pojmy:

**Proměnná** je pojmenovaná hodnota v paměti interpretovaná podle konkrétního datového typu.

**Příkaz** představuje spustitelnou část programu, pokud se provede jeden příkaz, tak se vyhodnotí všechny výrazy v něm.

Příkaz se může skládat z **výrazů**, ty se vyhodnotí teprve až se provede příkaz, ve kterém jsou použité.

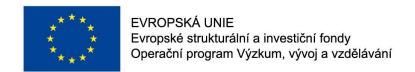
Například příkaz: int x = 5; rezervuje místo v paměti pro celé číslo, uloží tam hodnotu 5 jako množinu bitů a této hodnotě v paměti potom říkáme x. Typ int potom určuje, jaké operace (sčítání, odčítaní atd.) s proměnnou x může program provádět.

A nakonec máme **blok příkazů**, tedy více příkazů v jednom. V bloku příkazů se jednotlivé příkazy provádějí sekvenčně shora dolů. Blok příkazů je v jazyce C#.

# 1.2 Zásobník (Stack)

Každý program si alokuje po spuštění řádově megabajty paměti pro lokální proměnné, návratové adresy metod a případně další data. S touto pamětí se potom pracuje jako s datovou strukturou zásobník (Stack) a také se jí tak říká.

V následujícím příkladu si ukážeme příklad na zásobník, kdy vytvoříme několik lokálních proměnných a postupně je budeme přidávat nebo odebírat ze zásobníku.







## 1.3 Algoritmus výpočtu kořenů kvadratické rovnice

Konkrétně si ukážeme program pro výpočet kořenů kvadratické rovnice. Program využívá typ *double*, podmíněný výraz, výpis na konzoli s využitím *string interpolation* a statickou metodu *Math.Sqrt* pro výpočet druhé odmociny. Tyto vlastnosti jazyka probereme podrobně později. Příklad by měl být ale čitelný i pro čtenáře bez znalosti těchto vlastností.

```
using System;

double a = 1.0;
double b = 5.0;
double c = 6.0;

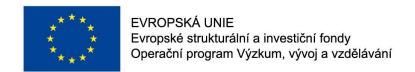
double D = (b * b) - (4 * a * c);

if (D > 0.0)
{
     double x1 = (-b + Math.Sqrt(D)) / (2 * a);
     double x2 = (-b - Math.Sqrt(D)) / (2 * a);

     Console.WriteLine($"x1: {x1} x2: {x2}");
}
```

Nejprve projdeme vlastní algoritmus výpočtu a poté ten samý program z hlediska alokace proměnných na zásobníku.

Kvadratická rovnice  $ax^2 + bx + c = 0$  je zadaná pomocí tří členů, kvadratického, lineárního a absolutního [1].







V programu pro ně používáme typ *double*, který představuje čísla s desetinou čárkou a znaménkem:

```
double a = 1.0;
double b = 5.0;
double c = 6.0;
```

Nejprve vypočítáme diskriminant pomocí následujícího aritmetického výrazu a výsledek uložíme do proměnné s názvem *D*:

```
double D = (b * b) - (4 * a * c);
```

S pomocí podmíněného příkazu if ověříme, zda je diskriminant *D* větší než *O* a pokud ano, tak se provede blok kódu ve složených závorkách:

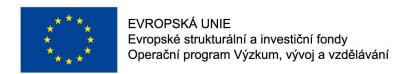
```
if (D > 0.0)
{
}
```

Konkrétně spočítáme dva kořeny kvadratické rovnice x1 a x2. Výraz Math.Sqrt(D) vrací druhou odmocninu proměnné D.

```
double x1 = (-b + Math.Sqrt(D)) / (2 * a);
double x2 = (-b - Math.Sqrt(D)) / (2 * a);
```

A výsledek vypíšeme na konzoli s využitím string interpolation  $x: x_1: x_2: x_2.$ K místo výrazů  $x_1: x_2: x_2.$ K místo výrazů  $x_1: x_2: x_2.$ 

```
Console.WriteLine($"x1: {x1} x2: {x2}");
```







Nakonec vypíšeme na konzoli hodnotu proměnné *D*. Tento výpis bychom mohli udělat i dříve a ne až na konci programu, ale díky výpisu na konci programu se bude lépe demonstrovat princip zásobníku a rozsah platnosti proměnné.

```
Console.WriteLine($"D: {D}");
```

### 1.4 Popis programu z hlediska zásobníku

Nyní si projdeme program z hlediska zásobníku. Popis zásobníku bude zjednodušený a zaměří se na základní princip zásobníku a lokálních proměnných. Připomínám, že uvedené adresy v paměti jsou ilustrační a ve skutečném programu budou jiné.

Na levé straně bude vždy uvedený kód a na pravé straně objekty v paměti RAM.

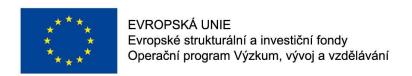
```
Kód RAM
using System;

double a = 1.0;
double b = 5.0;
double c = 6.0;

double D = (b * b) - (4 * a * c);

if (D > 0.0)
{
    double x1 = (-b + Math.Sqrt(D)) / (2 * a);
    double x2 = (-b - Math.Sqrt(D)) / (2 * a);

    Console.WriteLine($"x1: {x1} x2: {x2}");
}
Console.WriteLine($"D: {D}");
```







Pokud definujeme například proměnnou *a*, tak se pro ni alokuje místo pro typ *double* na zásobníku, například na adrese 1000.

```
Kód

using System;

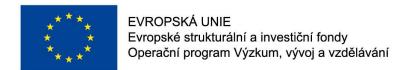
double a = 1.0;
    double b = 5.0;
    double c = 6.0;

double D = (b * b) - (4 * a * c);

if (D > 0.0)
{
    double x1 = (-b + Math.Sqrt(D)) / (2 * a);
    double x2 = (-b - Math.Sqrt(D)) / (2 * a);

    Console.WriteLine($"x1: {x1} x2: {x2}");
}

Console.WriteLine($"D: {D}");
```

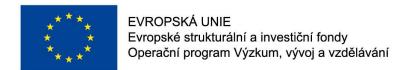






Protože typ double zabírá v paměti 8 bajtů, tak další proměnná b bude do paměti uložena hned za proměnnou a, tedy na adresu 1008.

Kód		RAM	
using System;			
<pre>double a = 1.0; double b = 5.0;</pre>	a 1000	1.0	
double c = 6.0;	b 1008	5.0	
double D = (b * b) - (4 * a * c);			
if (D > 0.0) {			
<pre>double x1 = (-b + Math.Sqrt(D)) / (2 * a); double x2 = (-b - Math.Sqrt(D)) / (2 * a);</pre>			
<pre>Console.WriteLine(\$"x1: {x1} x2: {x2}"); }</pre>			
<pre>Console.WriteLine(\$"D: {D}");</pre>			

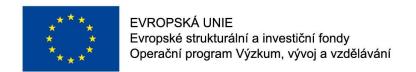






Stejným způsobem bude na zásobníku alokováno místo pro proměnnou c, která bude do paměti uložena hned za proměnnou b, tedy na adresu 1016.

Kód		RAM	
<pre>using System;</pre>			
<pre>double a = 1.0; double b = 5.0;</pre>	a 1000	1.0	
<pre>double c = 6.0;</pre>	b 1008	5.0	
<pre>double D = (b * b) - (4 * a * c); if (D &gt; 0.0)</pre>	c 1016	6.0	
{     double x1 = (-b + Math.Sqrt(D)) / (2 * a);			
double x2 = (-b - Math.Sqrt(D)) / (2 * a);			
<pre>Console.WriteLine(\$"x1: {x1} x2: {x2}"); }</pre>			
<pre>Console.WriteLine(\$"D: {D}");</pre>			

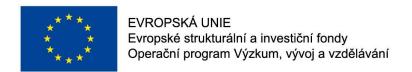






Další postup je stejný, pro proměnnou D opět alokujeme 8 bajtů na zásobníků, tentokrát na adrese *1024*.

Kód		RAM	
using System;			
<pre>double a = 1.0; double b = 5.0;</pre>	a 1000	1.0	
<pre>double c = 6.0;</pre>	b 1008	5.0	
double D = (b * b) - (4 * a * c);	c 1016	6.0	
<pre>if (D &gt; 0.0) {     double x1 = (-b + Math.Sqrt(D)) / (2 * a);</pre>	D 1024	1.0	
double x2 = (-b - Math.Sqrt(D)) / (2 * a);			
<pre>Console.WriteLine(\$"x1: {x1} x2: {x2}"); }</pre>			
<pre>Console.WriteLine(\$"D: {D}");</pre>			

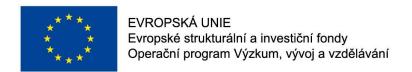






Další dvě proměnné x1 a x2 budou alokovány pouze pokud bude proměnná D mít hodnotu větší než 0.

Kód		RAM
using System;		
<pre>double a = 1.0; double b = 5.0;</pre>	a 1000	1.0
<pre>double c = 6.0;</pre>	b 1008	5.0
<pre>double D = (b * b) - (4 * a * c); if (D &gt; 0.0)</pre>	c 1016	6.0
<pre>double x1 = (-b + Math.Sqrt(D)) / (2 * a);</pre>	D 1024	1.0
double $x2 = (-b - Math.Sqrt(D)) / (2 * a);$	x1 1032	-2.0
<pre>Console.WriteLine(\$"x1: {x1} x2: {x2}"); }</pre>	x2 1040	-3.0
<pre>Console.WriteLine(\$"D: {D}");</pre>		

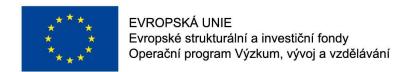






Až skončí blok, ve kterém byly definovány proměnné x1 a x2, tak budou tyto proměnné uvolněné z paměti zásobníku.

Kód		RAM
using System;		
<pre>double a = 1.0; double b = 5.0;</pre>	a 1000	1.0
double c = 6.0;	b 1008	5.0
<pre>double D = (b * b) - (4 * a * c); if (D &gt; 0.0)</pre>	c 1016	6.0
<pre>double x1 = (-b + Math.Sqrt(D)) / (2 * a);</pre>	D 1024	1.0
<pre>double x2 = (-b - Math.Sqrt(D)) / (2 * a);</pre>	x1 1032	-2.0
Console.WriteLine(\$"x1: {x1} x2: {x2}"); }	x2 1040	-3:0
<pre>Console.WriteLine(\$"D: {D}");</pre>		

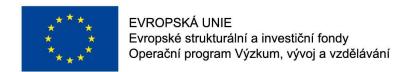






V zásobníku tedy zůstanou jen čtyři proměnné a poslední příkaz vypíše hodnotu proměnné D.

Kód		RAM
<pre>using System;</pre>		
<pre>double a = 1.0; double b = 5.0;</pre>	a 1000	1.0
<pre>double c = 6.0;</pre>	b 1008	5.0
double D = (b * b) - (4 * a * c);	c 1016	6.0
<pre>if (D &gt; 0.0) {     double x1 = (-b + Math.Sqrt(D)) / (2 * a);</pre>	D 1024	1.0
double x2 = (-b - Math.Sqrt(D)) / (2 * a);		
<pre>Console.WriteLine(\$"x1: {x1} x2: {x2}"); }</pre>		
<pre>Console.WriteLine(\$"D: {D}");</pre>		







Paměť pro proměnné a, b, c a d bude uvolněna na konci programu.

```
Kód RAM

using System;

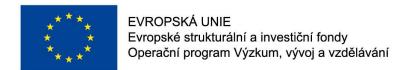
double a = 1.0;
double b = 5.0;
double c = 6.0;

double D = (b * b) - (4 * a * c);

if (D > 0.0)
{
    double x1 = (-b + Math.Sqrt(D)) / (2 * a);
    double x2 = (-b - Math.Sqrt(D)) / (2 * a);

    Console.WriteLine($"x1: {x1} x2: {x2}");
}

Console.WriteLine($"D: {D}");
```







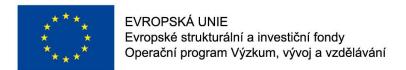
### 1.5 Zásobník a parametry metody

Na zásobník se také mohou ukládat parametry metod a návratové adresy metod, tedy adresa, od které má pokračovat provádění programu po ukončení metody. Problematika volání metod je obecně složitější a závisí na Calling Conventions (konvencích volání) a dalších optimalizacích [3]. V tomto materiálu budeme popisovat pouze **zjednodušené principy** pro pochopení základů.

Pokud bychom volali metodu rekurzivně, tedy metoda by neustále volala sama sebe, tak by časem došlo k přetečení zásobníku (stack overflow).

Následující příklad demonstruje, jakým způsobem se na zásobník ukládají parametry metody. Konkrétně budeme mít statickou metodu *Soucet*, která má dva parametry a vrací součet těchto parametrů.

Členské metody a statické metody probereme později, zatím nám budou složit jen pro demonstraci principu fungování zásobníku.

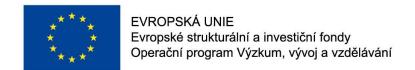






V příkladu tedy máme následující kód, který obsahuje metody *Soucet* a *Main*. Tentokrát nepoužijeme Top Level Statements [2], ale již zmíněnou metodu *Main*.

```
using System;
namespace CviceniZap
{
    class Program
        static int Soucet(int a, int b)
        {
            return a + b;
        }
        static void Main(string[] args)
        {
            int a = 2;
            int b = 3;
            int s = Soucet(a, b);
            Console.WriteLine($"soucet: {s}");
        }
    }
```







Metoda Soucet má dva parametry a, b typu celého čísla, tedy int. Tato metoda vrací součet těchto parametrů.

```
static int Soucet(int a, int b)
{
    return a + b;
}
```

Metoda *Main* definuje dvě lokální proměnné *a*, *b* typu *int*. Je důležité si uvědomit, že jde o jiné proměnné než parametry metody. Protože rozsah platnosti lokálních proměnných i parametrů metod je jen uvnitř bloků metod ohraničených složenými závorkami, mohou se tyto proměnné jmenovat stejně a přitom jde o jiné proměnné.

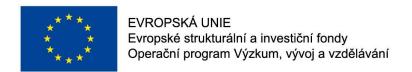
```
int a = 2;
int b = 3;
```

Nakonec v programu voláme metodu *Soucet* a předáváme jí hodnoty proměnných *a*, *b* jako argumenty. Návratová hodnota metody Soucet se potom přiřadí jako hodnota lokální proměnné *s*.

```
int s = Soucet(a, b);
```

Poslední příkaz Console.WriteLine(\$"soucet: {s}"); vypíše hodnotu proměnné s na konzoli.

Na dalších stránkách projdeme postupně jednotlivé příkazy z hlediska paměti a zásobníku. Z předchozího textu už by mělo být zřejmé jak se na zásobníku rezervuje automaticky paměť pro lokální proměné. Z příkladu bychom mohli vynechat alokaci prvních lokální proměnných, ale pro případ, kdyby chtěl někdo studovat tento příklad samostatně bude uvedena veškerá alokace proměnných postupně.

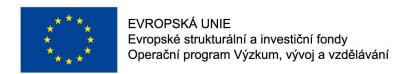






Zobrazení zásobníku je opět schématické a adresy proměnných v příkladu jsou vymyšlené. Skutečné adresy a alokace na zásobníku v reálném programu se může lišit. Na levé straně bude opět vždy uvedený kód a na pravé straně objekty v paměti RAM.

```
Kód
                                                              RAM
using System;
namespace CviceniZap
{
    class Program
   {
        static int Soucet(int a, int b)
            return a + b;
        }
        static void Main(string[] args)
        {
            int a = 2;
            int b = 3;
            int s = Soucet(a, b);
            Console.WriteLine($"soucet: {s}");
        }
   }
}
```

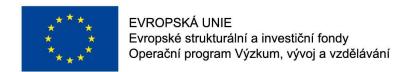






Nejprve definujeme paměť pro proměnnou *a* v kontextu metody *Main*, to znamená že se alokuje místo pro typ *int*, tedy celé číslo na zásobníku, například na adrese *1000*.

```
Kód
                                                               RAM
using System;
namespace CviceniZap
{
    class Program
                                                            Main: a
                                                                       2
                                                            1000
        static int Soucet(int a, int b)
            return a + b;
        }
        static void Main(string[] args)
        {
            int a = 2;
            int b = 3;
            int s = Soucet(a, b);
            Console.WriteLine($"soucet: {s}");
        }
   }
}
```

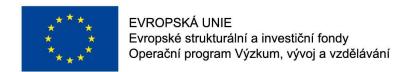






Protože typ int zabírá v paměti 4 bajty, tak další proměnná b bude do paměti uložena hned za proměnnou a, tedy na adresu 1004.

```
Kód
                                                               RAM
using System;
namespace CviceniZap
{
    class Program
                                                            Main: a
    {
                                                                        2
                                                             1000
        static int Soucet(int a, int b)
                                                            Main: b
                                                                        3
                                                             1004
            return a + b;
        }
        static void Main(string[] args)
        {
            int a = 2;
            int b = 3;
            int s = Soucet(a, b);
            Console.WriteLine($"soucet: {s}");
        }
    }
}
```

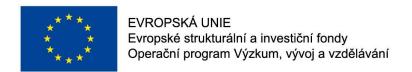






Při volání metody *Soucet* provede program skok na adresu kde jsou instrukce této metody. Aby potom věděl na jakou adresu skočit po ukončení metody, může si program uložit návratovou adresu na zásobník. Adresa zabere u *64* bitového programu *64* bitů, tedy *8* bajtů. Konkrétní implementace se ale může lišit dle platformy, překladače a souvisejících nastavení.

```
Kód
                                                                 RAM
using System;
namespace CviceniZap
{
    class Program
                                                              Main: a
                                                                          2
                                                               1000
        static int Soucet(int a, int b)
                                                              Main: b
                                                                          3
                                                               1004
             return a + b;
        }
                                                       Návratová adresa
                                                                        3000
                                                               1008
        static void Main(string[] args)
             int a = 2;
             int b = 3;
  Adresa 3000
           int s = Soucet(a, b);
             Console.WriteLine($"soucet: {s}");
        }
    }
}
```

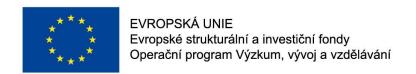






Nyní program alokuje paměť pro parametr *a* v kontextu metody *Main*, to znamená že se alokuje místo pro typ *int* a přiřadí mu hodnotu argumetu *a*. V reálném programu může být postup jiný, program může přimo využít registry, nebo další optimalizace a vše je závislé na platformě, překladači a dalších souvisejících nastaveních. Z hlediska principu z pohledu programovacího jazyka je pro nás duležíté, že získáváme **kopii** hodnoty.

```
Kód
                                                                  RAM
using System;
namespace CviceniZap
{
    class Program
                                                               Main: a
    {
                                                                           2
                                                               1000
        static int Soucet(int a, int b)
                                                               Main: b
                                                                           3
                                                               1004
             return a + b;
        }
                                                        Návratová adresa
                                                                         3000
                                                               1008
        static void Main(string[] args)
                                                             Soucet: a
                                                                           2
                                                               1016
             int a = 2;
             int b = 3;
             int s = Soucet(a, b);
             Console.WriteLine($"soucet: {s}");
        }
    }
}
```

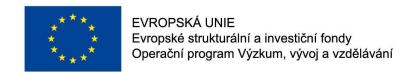






Nyní program alokuje stejným způsobem paměť pro parametr b v kontextu metody Main a přiřadí mu hodnotu argumentu b, tedy hodnotu 3.

```
Kód
                                                                  RAM
using System;
namespace CviceniZap
{
    class Program
                                                              Main: a
                                                                          2
                                                               1000
        static int Soucet(int a, int b)
                                                              Main: b
                                                                          3
                                                               1004
             return a + b;
        }
                                                       Návratová adresa
                                                                         3000
                                                               1008
        static void Main(string[] args)
                                                             Soucet: a
                                                                          2
        {
                                                               1016
             int a = 2;
                                                            Soucet: b
             int b = 3;
                                                                          3
                                                               1020
             int s = Soucet(a, b);
             Console.WriteLine($"soucet: {s}");
        }
    }
}
```

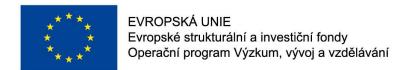






Výsledek součtu proměnných si program pro použití příkazu *return* uloží do registru procesoru a vrátí se na návratovou adresu aby pokračoval v metodě *Main*.

Kód	RA	M
using System;		
namespace CviceniZap		
{     class Program     {	Main: a 1000	2
<pre>static int Soucet(int a, int b) {     return a + b;</pre>	Main: b 1004	5.0
}	Návratová adresa 1008	3000
<pre>static void Main(string[] args) {</pre>	Soucet: a 1016	2
<pre>int a = 2; int b = 3;</pre>	Soucet: b 1020	3
→ int s = Soucet(a, b);		
<pre>Console.WriteLine(\$"soucet: {s}"); }</pre>		
}		

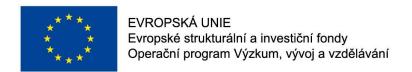






Parametry a, b a návratová adresa se po skončení metody Soucet uvolní z paměti.

```
Kód
                                                                 RAM
using System;
namespace CviceniZap
{
    class Program
                                                              Main: a
    {
                                                                          2
                                                               1000
        static int Soucet(int a, int b)
                                                              Main: b
                                                                         5.0
                                                               1004
             return a + b;
        }
                                                       Návratová adresa
                                                                         3000
                                                               1008
        static void Main(string[] args)
                                                            Soucet: a
                                                               1016
        {
             int a = 2;
                                                            Soucet: b
                                                                          3
            int b = 3;
                                                               1020
            int s = Soucet(a, b);
            Console.WriteLine($"soucet: {s}");
        }
    }
}
```

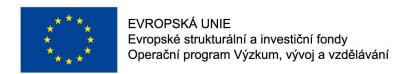






Na zásobníku se potom rezervuje místo pro proměnnou s do které se uloží návratová hodnota vrácená z programu Soucet příkazem return (dočasně uložená v registru procesoru).

```
Kód
                                                                RAM
using System;
namespace CviceniZap
{
    class Program
                                                            Main: a
    {
                                                                        2
                                                             1000
        static int Soucet(int a, int b)
                                                            Main: b
                                                                        5.0
                                                             1004
            return a + b;
        }
                                                            Suma: s
                                                                        5
                                                             1016
        static void Main(string[] args)
        {
            int a = 2;
            int b = 3;
            int s = Soucet(a, b);
            Console.WriteLine($"soucet: {s}");
        }
    }
}
```

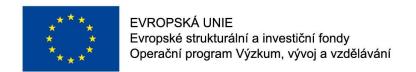






V závěru potom program vypíše hodnotu proměnné s na konzoli.

```
Kód
                                                                RAM
using System;
namespace CviceniZap
{
    class Program
                                                            Main: a
    {
                                                                         2
                                                             1000
        static int Soucet(int a, int b)
                                                            Main: b
                                                                        5.0
                                                             1004
            return a + b;
        }
                                                           Soucet: s
                                                                         5
                                                             1000
        static void Main(string[] args)
        {
            int a = 2;
            int b = 3;
            int s = Soucet(a, b);
            Console.WriteLine($"soucet: {s}");
        }
    }
}
```

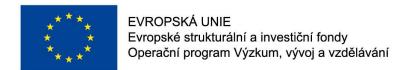






Po ukončení metody *Main* dojde k uvolnění všech lokálních proměnných, které byly v metodě *Main* alokovány.

```
Kód
                                                             RAM
using System;
namespace CviceniZap
{
   class Program
        static int Soucet(int a, int b)
            return a + b;
        }
        static void Main(string[] args)
        {
            int a = 2;
            int b = 3;
            int s = Soucet(a, b);
            Console.WriteLine($"soucet: {s}");
   }
}
```







# SEZNAM POUŽITÉ LITERATURY

- [1] Kvadratické rovnice Matematika.cz. *Matematika pro střední a základní školy Matematika.cz* [online]. Copyright © 2006 [cit. 10.12.2020]. Dostupné z: https://matematika.cz/kvadraticke-rovnice
- [2] What's new in C# 9.0 C# Guide | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 22.12.2020]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-9#top-level-statements
- [3] Calling Conventions | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 01.01.2021]. Dostupné z: https://docs.microsoft.com/en-us/cpp/cpp/calling-conventions?view=msvc-160

