Profiling – shrnutí

Monitorování programu bylo provedeno pomocí nástroje Profiling Tools integrovaném ve Visual studiu. Pro lepší analýzu jsme četli ze souboru, kde byly vygenerovány náhodné kombinace čísel.

Analýza programu:

Metoda	Počet volání	Čas strávený ve funkci (%)	
profiling.Program.Main(string[])	1	100,00	7,03
System.Console.WriteLine(strin	50	72,94	72,94
System.IO.StreamReader..ctor(s	1	2,57	2,57
System.Double.Parse(string)	2 660	2,22	2,22
MathLib.CalcMath.Root(float64	50	1,82	1,82
System.String.Concat(string,stri	7 715	1,41	1,41
MathLib.CalcMath.Subtract(floa	5 320	1,35	1,35
MathLib.CalcMath.Add(float64,	5 320	1,33	1,33
System.Char.IsDigit(char)	20 750	1,24	1,24
System.IO.TextReader.ReadLine	51	1,16	1,16
MathLib.CalcMath.Multipy(float	2 660	1,09	1,09
System.String.get_Length()	18 140	1,08	1,08
System.Char.ToString()	7 715	1,02	1,02
MathLib.CalcMath.Divide(float6	100	0,99	0,99
MathLib.CalcMath..ctor()	1	0,88	0,88
System.Char.IsWhiteSpace(char	13 035	0,79	0,79
System.String.get_Chars(int32)	10 375	0,58	0,58
System.Double.ToString()	50	0,33	0,33
System.IO.TextReader.Close()	1	0,16	0,16
System.String.op_Inequality(str	51	0,03	0,03

Podle předpokládání tráví program nejvíce času, co se týče matematické knihovny, v metodě Root (odmocnina). Metoda sama o sobě není robustní, ale volá metodu Pow (mocninu) a dělí. Myslíme, že pro optimalizaci zde by bylo nejlepší, ne se soustředit na metodu Root, ale na Pow. Optimalizací Pow bychom dosáhli lepšího výsledku v obou metodách.

Další je odčítání a dělení, to je hlavně proto, že jsou volány mnohokrát více, než Root. Samozřejmě optimalizováním jen o trochu bychom mohli dosáhnout vzhledem k počtu volání velké změny.
V normálních případech je to ovšem zbytečné, navíc není téměř co optimalizovat.