



# IVS – Profilovanie programu

Projekt: Výpočet výberovej smerodajnej odchýlky

**Tím:**

OOP haters

Dávid Vihonský – xvihond00

Ľubomír Durkáč – xdurkal00

Pavol Mihálik – xmihalp01

29. apríla 2025

# Obsah

<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Použité nástroje</b>	<b>2</b>
<b>3</b>	<b>Vizualizácia profilovania</b>	<b>3</b>
3.1	Vstup: 10 čísel . . . . .	3
3.2	Vstup: 1000 čísel . . . . .	5
3.3	Vstup: 1 000 000 čísel . . . . .	6
<b>4</b>	<b>Analýza výsledkov</b>	<b>7</b>
<b>5</b>	<b>Možnosti optimalizácie</b>	<b>7</b>
<b>6</b>	<b>Záver</b>	<b>7</b>

# 1 Úvod

Cieľom tejto úlohy bolo vytvoriť program na výpočet výberovej smerodajnej odchýlky zo vstupnej postupnosti čísel. Program bol napísaný v jazyku C++ a profilovaný pomocou nástrojov **gprof** a **Callgrind**. Profilovanie prebiehalo pre vstupy rôznej veľkosti: 10, 1000 a 1 000 000 čísel.









# 2 Použité nástroje

Profilovanie bolo realizované pomocou nasledujúcich nástrojov: **gprof** (základné profilovanie na úrovni funkcií), **valgrind --tool=callgrind** (detailná analýza počtu inštrukcií), **KCachegrind** (vizuálne zobrazenie volaní funkcií a ich záťaže).

## 3 Vizualizácia profilovania

### 3.1 Vstup: 10 čísel

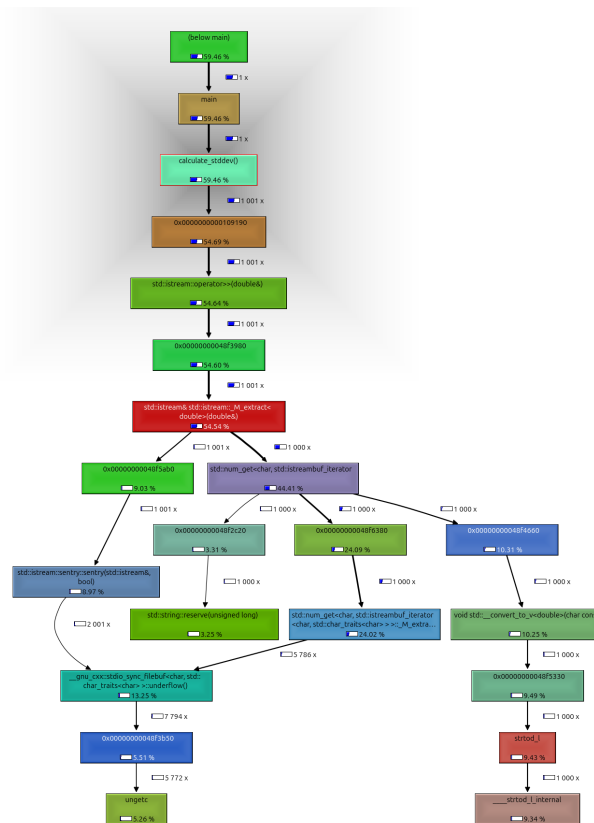
Search: calc ✕ (No Groupin

Incl.	Self	Called	Function	Location
2.03	0.03	1 	calculate_stddev()	stddev: stddev.cpp
0.05	0.01	5 	Calculator::power(doubl...	stddev: mathlibrary.cpp
0.06	0.01	20 	Calculator::add(double, ...	stddev: mathlibrary.cpp
0.06	0.01	1 	Calculator::root(double,...	stddev: mathlibrary.cpp
0.02	0.01	6 	Calculator::isInteger(do...	stddev: mathlibrary.cpp
0.04	0.01	12 	Calculator::mul(double, ...	stddev: mathlibrary.cpp
0.01	0.00	2 	Calculator::div(double, d...	stddev: mathlibrary.cpp
0.00	0.00	1 	Calculator::sub(double, ...	stddev: mathlibrary.cpp



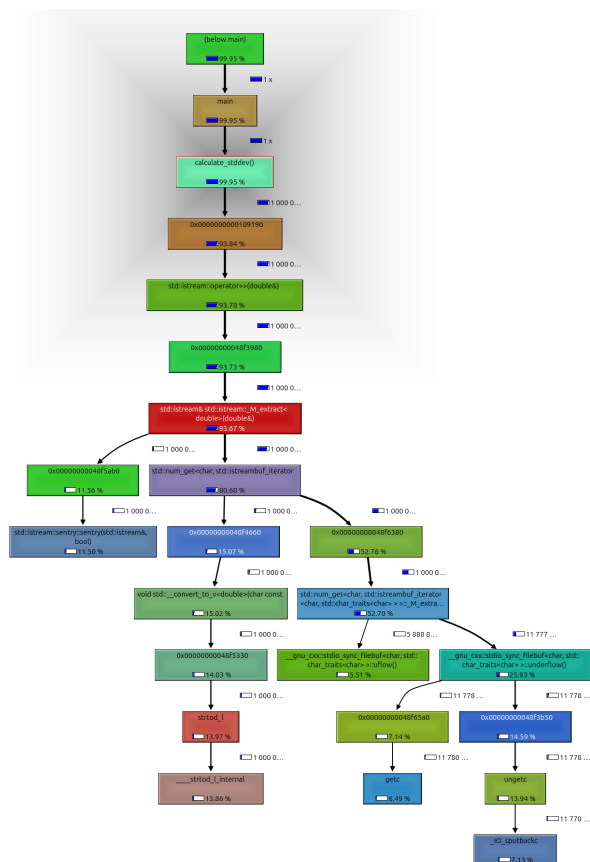
### 3.2 Vstup: 1000 čísel

Incl.	Self	Called	Function	Location
59.46	0.83	1	calculate_stddev()	stddev: stddev.cpp
2.50	0.51	2 000	Calculator::add(double, ...	stddev: mathlibrary.cpp
1.25	0.25	1 002	Calculator::mul(double, ...	stddev: mathlibrary.cpp
0.06	0.01	1	Calculator::root(double, ...	stddev: mathlibrary.cpp
0.05	0.02	13	Calculator::power(doubl...	stddev: mathlibrary.cpp
0.02	0.01	14	Calculator::isInteger(do...	stddev: mathlibrary.cpp
0.00	0.00	2	Calculator::div(double, d...	stddev: mathlibrary.cpp
0.00	0.00	1	Calculator::sub(double, ...	stddev: mathlibrary.cpp



### 3.3 Vstup: 1 000 000 čísel

Incl.	Self	Called	Function	Location
99.95	1.07	1	calculate_stddev()	stddev: stddev.cpp
3.25	0.66	2 000 000	Calculator::add(double, ...	stddev: mathlibrary.cpp
1.62	0.33	1 000 002	Calculator::mul(double, ...	stddev: mathlibrary.cpp
0.00	0.00	1	Calculator::root(double, ...	stddev: mathlibrary.cpp
0.00	0.00	23	Calculator::power(doubl...	stddev: mathlibrary.cpp
0.00	0.00	24	Calculator::isInteger(do...	stddev: mathlibrary.cpp
0.00	0.00	2	Calculator::div(double, d...	stddev: mathlibrary.cpp
0.00	0.00	1	Calculator::sub(double, ...	stddev: mathlibrary.cpp



## 4 Analýza výsledkov

Z výstupov gprof aj Callgrind je zrejmé nasledovné:

- Pri malom počte vstupov je výpočtová záťaž zanedbateľná, strom volaní je malý a prehľadný.
- Pri 1000 vstupoch sa začínajú objavovať vyššie nároky na vstupno-výstupné operácie, no stále je výpočet dominantný.
- Pri 1 miliónu vstupov dominuje čítanie vstupov cez `std::cin`, ktoré tvorí väčšinu vykonaných inštrukcií.
- Funkcia `calculate_stddev()` volá najčastejšie `add()` a `mul()`, čo je prirodzené pri výpočte priemeru a odchýlky.
- Výpočtové funkcie z knižnice sú efektívne, ale čítanie dát predstavuje najväčší problém z hľadiska výkonu.

## 5 Možnosti optimalizácie

- Nahraďme pomalé čítanie `std::cin` za `getline()` + `std::istream`, čo by znížilo počet volaní I/O operácií.
- V prípade veľmi veľkých dát by pomohlo načítanie blokov naraz do pamäte.
- Výpočet súčtov a priemerov by mohol byť rozdelený do paralelných častí napríklad pomocou OpenMP.

## 6 Záver

Profilovanie potvrdilo, že výpočtová časť programu je efektívna, avšak pri veľkých dátach sa ako hlavný problém ukázalo pomalé načítavanie vstupov. Napriek tomu program zvláda aj milión vstupov bez problémov a jeho výstupy sú správne. S navrhnutými optimalizáciami by bolo možné ešte výraznejšie zlepšiť výkon pri spracovaní veľkých súborov.