Compare Increment

Hao Deng

October 2023

1 Introduction

In the last week, we compared the postfix and prefix increment on different datatypes in C++, including:

1. int32_t
2. int64_t
3. uint32_t
4. uint64_t
5. float
6. double

Theoretically, we expect the prefix increment should have less time cost because the postfix increment potentially makes a copy of the original data. However, the result showed little difference between the time cost of postfix and prefix increment. We speculated that modern compilers can automatically optimize the postfix increment.

In this assignment, we want to address the problem left in the previous one and optimize our code meanwhile. We propose to use the template to make our code more concise and robust. To prevent compilers from automatically optimizing the postfix increment, we need to overload both the postfix and the prefix increment. In this assignment, we decided to fix the number of iterations at 2,000,000,000 times.

2 Experiment Result

	int32_t	int64_t	uint32_t	uint64_t	float	double
i++	6.077s	6.066s	6.07911s	5.82797s	6.27s	6.358s
++i	3.845s	3.555s	3.56102s	3.79405s	4.18s	4.15897s

Table 1: Experiment 1 with 2,000,000,000 Iterations

	int32_t	int64_t	uint32_t	uint64_t	float	double
i++	6.19555s	6.07302s	6.121s	6.392s	6.283s	6.50402s
++i	3.705s	3.71298s	3.704s	3.733s	4.44168s	4.37302s

Table 2: Experiment 2 with 2,000,000,000 Iterations

	int32_t	int64_t	uint32_t	uint64_t	float	double
i++	5.94798s	5.961s	5.983s	6.021s	6.276s	6.492s
++i	3.66802s	3.63s	3.6551s	3.65302s	4.37702s	4.43698s

Table 3: Experiment 3 with 2,000,000,000 Iterations

3 Analysis

From the tables above, we can readily notice that prefix increment is faster than postfix increment. Therefore, we should expect using prefix increment can improve the performance of codes.