**Spike:** 2

**Title:** Performance Measurement

**Author:** Ryan Chessum, 102564760

**Goals / deliverables:**

* Code, see /07 – spike - Performance Measurement/Performance Measurement/
* Spike Report

**Technologies, Tools, and Resources used:**

List of information needed by someone trying to reproduce this work

* Visual Studio 2019
* C plus plus reference (<https://www.cplusplus.com/reference/>)
* Canvas sample code

**Tasks undertaken:**

* Download and install Visual Studio
* Create a new C++ project

**What we found out:**

Performance measurement is.

We can measure the time it takes to measure a function by recording the time before we star doing anything, performing the tasks we want to record time for, recording the time again afterwards and then getting the difference between the two recorded times.

A screenshot of a computer

Description automatically generated with medium confidence

Ramp tests change the variable size to determine how the operation will perform with different variable sizes. Ramp up tests increase the size while ramp down tests decrease it.

An exponential ramping test changes the size in exponential increments. This can help us notice difference in performance that may not be clear at smaller when we change the size linearly.

Using the test code provided I ran each test 3 times and recorded some data on linear and exponential ramp up and down tests. I’ve recorded multiple times for the same size as the time can vary. The computer is running other tasks at the same time so depending on how intensive those tasks are the time to execute will be longer or shorter.

Table

Description automatically generated

It’s only a small data size but we can see that at each size we are getting roughly the same time with no outliers that are much bigger than the other times.

If we graph the data, we can have a look at how the time taken to perform the function changes with the size of the vector.

Chart, scatter chart

Description automatically generated

Looking at both charts for the linear ramp tests it seems like the time it takes to execute the function increases linearly. It also gives us a good look at the outliers compared to the linear trend.

However, the increments we are using are quite small. Sometimes we may find that with larger values the time it takes to complete tasks will become much bigger as they get more complex.

If we look at the charts for the exponential ramp tests, we can see that the relationship between vector size and function execution time is still linear. Though the time and size get so big that we can’t really see the other smaller points of data. That’s ok though as we can confirm that the execution time increases linearly. So, if we wanted to look into the variance we could test a different set of data that would help us find that.

Chart, line chart

Description automatically generated

We can use these ideas to compare the performance of different functions as well.

Let’s compare these two functions.

Text

Description automatically generated

I tested each one with 3 different string sizes to get some sample data.

Table

Description automatically generated

The time for each size seem pretty random when looking at them by eye.

Chart

Description automatically generated

If we look at the first of results chart, we can see some pretty interesting stuff though. To start off the data isn’t very varied at all. But as the string size gets bigger the time becomes a lot more varied. The trendline suggests that as the size increases, the possible execution time becomes higher.

Chart, scatter chart

Description automatically generated

The using count chart also indicates that variance occurs as the sting size increases. However, it doesn’t increase by nearly as much. The trendline stays mostly flat.

It’s also interesting to note the two outliers in both charts. For some reason that time it took way longer to execute. There must have been something else running in the background which caused it to take longer than usual. Perhaps the outliers are also slightly skewing the trendline? If we wanted to find out we would need a larger sample size.

A screenshot of a computer screen

Description automatically generated with medium confidence

Another thing that can cause execution time to take longer is using debug mode vs release mode. Debug mode is helpful as it can help us catch errors in our code as we’re writing it. But these checks take time to process so they increase the execution time. We can easily see the difference by switching to release mode and comparing the values.

Debug mode:

Text

Description automatically generated

Release Mode:

Text

Description automatically generated

The times are much faster.

Another thing that affects execution is compiler optimisation. When code is compiled, it often has optimizations made by the compiler to improve performance. If we were to turn these optimizations off the execution time would become much slower.

Release mode (Optimization):

Text

Description automatically generated

Release mode (No optimization):

Text

Description automatically generated

The times with no optimization are much slower,

To change optimizations in Visual Studio right click on the project and select properties. Then go to: Configuration Properties -> C/C++ -> Optimization. Then under the optimization tab select the optimization setting you want from the dropdown menu.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated