Week 2 Project—Syntax and Runtime Management of Dynamic Structures

**Scenario**

In this week’s lab you will look at timing operations and how different operations can take a different amount of time to complete.

**Rubric**

Point distribution for this activity:

|  |  |  |
| --- | --- | --- |
| **Project Activity** | | |
| **Document** | **Points possible** | **Points received** |
| Part A | 30 |  |
| Part B | 20 |  |
| **Total Points** | **50** |  |

PART A:

We will look at timing operations in C++. To see the difference in operations, write C++ code to compare cout and printf and display the time difference for 100 cout operations and 100 printf operations. This code can be written in Visual Studio. Below is how you would time the 100 cout operations in Visual Studio. Add another for loop and display time for both cout and printf then comment about why you think there is a difference.

#include <iostream>

#include <time.h>

using namespace std;

int main()

{

double start, stop;

start=clock();

for (int i=0;i<100;i++)

{

cout<<" The number is "<<i<<endl;

}

stop =clock();

cout<<"It took "<<(double(stop-start)/CLOCKS\_PER\_SEC)<<" seconds for cout"<<endl;

cin.ignore();

}

C++ code:

double startTwo, stopTwo;

startTwo = clock();

for (int x = 0; x < 100; x++)

{

printf(" The number is %d \n", x);

}

stopTwo = clock();

cout << "It took " << (double(stop - start) / CLOCKS\_PER\_SEC) << " seconds for cout" << endl;

cout << "It took " << (double(stopTwo - startTwo) / CLOCKS\_PER\_SEC) << " seconds for printf" << endl;

cin.ignore();

Screenshot of output:

A screen shot of a computer

Description automatically generated

Why is one faster than the other?

This boils down to the differences in how both of these commands function, with printf being a format based print statement and cout being a user specified print statement that needs the user to overload each step. But I did find this <https://stackoverflow.com/questions/2872543/printf-vs-cout-in-c> that explains that each << is another set of arguments that must be compared and processed, slowing down the process, where printf only has the one set of arguments passed.

Try this code in Java. Use eclipse or netbeans to see the difference in operations. Add for loops and display time for both System.out.pritnln and System.out.printf then comment about why you think there is a difference

**public** **class** optMain {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**long** start, stop=0;

start = System.*nanoTime*();

**for** (**int** i=0;i<100;i++)

{

System.***out***.println(" The number is "+i);

}

stop = System.*nanoTime*()-start;

System.***out***.println("It took "+stop+" nano seconds for code to run");

}

}

Java code:

Screenshot of output:

Why is one faster than the other?

Part B:

Next we will take timing a step further. Feel free to use C++, C#, or Java (ArrayList) for this next exercise. Determine the runtime differences between a fixed array and a dynamic array. Add several items to each array (I added 100000). A fixed array is created in the stack and allocated at compile time whereas a dynamic array is created in the heap and allocated at run time.

Copy your C++ or Java Code here:

Fixed array time: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (this may be 0)

Dynamic array time: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Explain the difference between the stack and the heap below:

Example output

