

# C++ Programming: Think Before You Code

Nthikeng Letsoalo

University of the Witwatersrand — August 25, 2018

## Introduction

Today's lab will focus primarily on your problem solving skills. The idea is to get you thinking about a problem before you actually start to write code. After having solved the problem using pen and paper (it is highly recommended that you do this), it is time to translate your thoughts into code. When solving your problem remember to use key words like **if**, **for**, **while** etc. during your thought process, this will guide your intuition when translating your thoughts to code which is the main objective.

For today's lab I will not be giving you any skeleton code. With that being said, I expect you to create your own functions with whatever parameters you see fit and return appropriate values if it is necessary (remember that you can have *void* functions)

## 1 Compiling Your Code

To compile and run your code follow the following structure:

Command Line

```
$ g++ -std=c++11 yourfile.cpp -o yourExecutable  
$ ./yourExecutable
```

## 2 Problem: Fizz - Buzz

To play this game players generally sit in a circle, However in your case you will think of having  $n$  players sitting in a straight line. The player designated to go first says the number "1", and each player thenceforth counts one number in turn. However, any number divisible by 3 is replaced by the word fizz and any divisible by 5 is replaced by the word buzz. Numbers divisible by both become fizz-buzz.

### 2.1 Consider $n$ players in a straight line

1.) Write a program that would simulate this game, your program should take in a value  $n$  and give the correct output from 1 to  $n$ :

input: 16

output: 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, Fizz Buzz, 16

### 2.2 Consider $n$ players in a circle

Can you write a simulation of this game to represent players sitting in a circle? Given  $n$  as the number of people in the game and  $r$  as the number of rounds of a game. If we have 4 players and we wish to play 4 round then  $n=4$  and  $r=4$  then the output should look like this:

Round 1: 1, 2, Fizz, 4  
Round 2: Buzz, Fizz, 7, 8  
Round 3: Fizz, Buzz, 11, Fizz  
Round 4: 13, 14, Fizz Buzz, 16

### 3 Terminal Illustrations

### 3.1 Christmas Tree

Given a height  $h$  are you able to draw a Christmas tree? If you are given a height  $h=5$ , print the following Christmas tree to the terminal, the base of the tree always has a width of 3 starts and a height of 2. Thus in total the height of your tree should be:  $h+2$

```
nthikeng@DellPC: ~/Desktop/Wits/ExtraLessons/Lab2/Solutions/ChristmasTree
nthikeng@DellPC:~/Desktop/Wits/ExtraLessons/Lab2/Solutions/ChristmasTree$ ./tree
Enter Tree height: 5
  *
 ***
*****
*****
*****
  ***
  ***
nthikeng@DellPC:~/Desktop/Wits/ExtraLessons/Lab2/Solutions/ChristmasTree$ _
```

Figure 1: Christmas Tree

### 3.2 Creating an Information Board\*

If you go to the Wits Bus Stop, you will notice that there is an information board that tells us the various time for departing buses. We will try to recreate some basic functionality of the board..

You are given a text that has *information-board-letters* that are separated by a # , you are required to take an upper case word as input and print it to the terminal using the alphabets that are found in the text file.


The figure below show the expected input and output

```

nithikeng@DellPC: ~/Desktop/Wits/ExtraLessons/Lab2/Solutions/InformationBoard
nithikeng@DellPC:~/Desktop/Wits/ExtraLessons/Lab2/Solutions/InformationBoard$ ./s
.o
NTHIKENG
s
* * * * * * * * * * * * * * * *
** * * * * * * * * * * * * *
V * * * * * * * * * * * * *
* * * * * * * * * * * * *
n * * * * * * * * * * * * *
nithikeng@DellPC:~/Desktop/Wits/ExtraLessons/Lab2/Solutions/InformationBoard$ _

```

Figure 2: Information board

 **Warning:** This question is not for the feint hearted

## 4 Sieve of Eratosthenes

This is an ancient algorithm developed by Eratosthenes to find prime numbers. It works by finding the lowest prime number then discarding all the multiples of that prime. It will do so iteratively until there are no prime numbers left in the list.

The sieve of Eratosthenes stops when the square of the number we are testing is greater than the last number on the grid (in our case 100).

Since  $11^2 = 121$  and  $121 > 100$ , when we get to the number 11, we can stop looking.



Figure 1: Sieve of Eratosthenes