LW: Passing by Reference

# Overview

## Objectives

* Understanding how arguments are passed to functions.
* Use passing by reference.
* Use passing by pointer.

## Introduction

Changing the value of an argument within the function will (generally) not affect the value of the argument passed in after execution of the function. This behavior is referred to as **pass-by-value**, as copies of the arguments are made for use within the function. Essentially, in pass-by-value, the variable within the function is initialized with the value passed into the function. For example, an integer foo can be passed by value simply by writing int foo in the argument list.

In contrast, **pass-by-reference** does not copy the argument; it actually hands the function the original memory associated with the argument. In C++, we default to pass-by-value, but we can tell the compiler to use pass-by-reference by appending an ampersand (&) to the data type. For example, to pass an integer goo by reference, we would place int& goo in the argument list. In pass-by-reference the function uses the exact same variable as the one passed in, except it now might have a different name or nickname.

C++ also inherits from the C programming language the ability to **pass-by-pointer**. By using a pointer type, we can access the memory locations of variables that exist outside the function. This is really a special case of pass by value, as changing where the variable points to will not persist outside the function. Only dereferencing the pointer and changing the value at that memory location will persist. To add an argument foo that points to an integer, we would place int\* foo in the argument list. An integer can be passed to this function by using the address operator (&). For example, an integer goo can be passed as a pointer using &goo.

## Getting Started

* Download the [starter code](https://drive.google.com/file/d/1SCY_YFRTGiQ6jqYHygkyFPvbOKlnYpZc/view?usp=share_link).
  + main.cpp
    - Contains a test program.
  + function.cpp
    - Contains the function definitions.
  + functions.h
    - Contains the function prototypes.
* Read through the introduction.
* Read through and complete the [requirements](#_sgvm56687jdu).
* **You need 40 / 40 points on Gradescope for completion**.
* Do not add any include statements.

# Requirements

## function\_one() - Pass by Reference

* Implement function\_one() and its prototype in functions.h.
  + Should accept two arguments, i and j.
  + Should add 2 to i and add 1 to j.
  + Changes to j should persist after the function, but not changes to i.
* Example:
  + If i = 3 and j = 4 are passed to function\_one(i, j), then after execution i = 3 and j = 5.

## function\_two() - Pass by Reference with Struct

* Implement function\_two() and its prototype in functions.h.
  + Should accept an argument of type struct example.
    - Struct is defined in functions.h.
  + Should increment the value of the struct by 1.
  + This change should persist after the function.
* Example:
  + If a struct e of type example with value = 4 is passed into function\_two(e), then after execution e.value = 5.

## function\_three() - Pass by Pointer

* Implement function\_three().
  + Function accepts a pointer to an integer, k, and an integer, l.
  + Should increment both values.
  + Changes to both k and l should persist after the function.
* **Do not modify the function prototype / function arguments**.
* **You will also need to add a function call to this function in main.cpp**.
  + On line 59 in main.cpp of the starter code, uncomment the function call and pass a and b appropriately.
* Example:
  + If an integer k = 5 and an integer l = 10 were passed into function\_three() with the correct syntax, then after execution k = 6 and l = 11.

## function\_four() - Passing an Array

* Implement function\_four() and its prototype in functions.h.
  + Should accept an integer array, an integer size (indicating the size of the array), and two integers called lowest and highest.
  + Function should iterate through the array and increment every value by two.
  + Function should set lowest and highest to the min / max of the array (after being incremented).
  + Changes to the array, lowest, and highest should all persist after the function.
* Example:
  + If an array {0, 1, 2, 3, 4} were passed into function\_four() along with size = 5 and integers lowest and highest, then after execution array should be {2, 3, 4, 5, 6}, lowest should be 2 and highest should be 6.