Hey everyone! This week we will continue our talk about functions. This is an expansion on last week's material so make sure to go read last week's resource if you have not already. Just as a reminder: These documents are intended for going over concepts that are discovered in class, as they are a concise and to the point review of the materials. But remember, these documents are not a replacement for lecture and they cannot cover everything, so make sure you get help on any specific concepts you struggle with as well as go over lecture notes, programs, and the textbook.

Reminder: Our group tutoring sessions will be held on Thursdays from 7:15PM - 8:15PM, in Cashion C202. You can get more information and reserve a spot at https://baylor.edu/tutoring.

**Keywords:** Functions, Pass By Value, Pass by Reference

---

**TOPIC OF THE WEEK**

*Pass by Value vs. Pass by Reference*

Last week we defined a function header where one of the key components of the header was the return type from the function. We noted that there are some instances where we do not want to return from the function and will thus use the “void” specifier. However, we might want to be able to change a variable just by passing it to a function rather than setting it equal to the return of a function. For this we need to understand the concept of **Pass by Value** vs. **Pass by Reference**.

**Pass by Value:**

Normally when you pass a primitive variable (int, double, etc.) to a function, you are by default passing the parameter value to the function by value. Passing a
variable to a function by value essentially makes a copy of the variable’s “value” and does everything in that function with the copy of the value that has been generated. Therefore any changes that have been made to that specific copy value will not be saved and will only be changed for as long as the copy is within the scope of the function.

If we take this simple program for example. The intent of the function is to double the value of the number variable (which starts at a value of 10). However, according to the logic of the definition above, passing by value will have no effect on the value of the number variable. Therefore, the value printed will be unaffected and still remain as 10.

Pass by Reference:

If we would like to actually be able to change the value passed to a function, then we need to invoke the use of pass by reference. Passing data to a function by reference can be triggered by using ‘&’. This allows us to change the actual value of a variable being passed to a function. Rather than making a copy of the value and passing that value to the function like before, this way will pass a reference to the ACTUAL variable and thus any modifications done to the variable used within the function, will be made to the function back in the main() function.

In our example of doubling the value of the number variable, to correctly do such we have to include the ‘&’ like mentioned before by appending it to the data type. In this case the variable is integer so we specify int& to represent passing an int by reference. Therefore, since we are now passing the variable by reference and therefore essentially passing the
actual variable itself, then the value displayed should now be the intended value of 20.

---

CHECK YOUR LEARNING

1. What is the specifier that needs to be included with a variable type when passing by reference?
2. If we pass a value by reference and change it within the instance of the function, will it be changed once the function is stepped out of? If so, why?

---

THINGS STUDENTS MAY STRUGGLE WITH

1. Comparing passing by reference to how we originally did it with “Returning from a function” last week, there is no real benefit or difference on which you choose. It often comes down to preference and design of how you want to do things. Take a look at the example of how we can achieve this same behavior with returning a value.

```cpp
#include <iostream>
using namespace std;

int doubleValue(int numByVal);

int main() {
    int number = 10;
    number = doubleValue(number);
    cout << number << endl;
    return 0;
}

int doubleValue(int numByVal) {
    return numByVal * 2;
}
```
Answers:

1. &
2. No, it will not be changed. Passing by value makes a copy of the variable's value to be used within the instance of the function and will not alter the original variable's value.

Note: All tables were taken from the Baylor CS 1430 zyBook