Week 14

Intro to Computer Science - CSI 1430

Hello and welcome to the weekly resources for CSI 1430: Introduction to Computer Science! This week's resource will be covering the course material that will be covered in week 14 of the course.

It is important to understand the difference between passing a variable by value and by reference—this resource seeks to clarify that distinction. **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. However, they are not a replacement for lecture and 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: If you have any questions about these study guides, group tutoring sessions, private 30-minute tutoring appointments, the Baylor Tutoring channel or any tutoring services we offer, please visit our website https://baylor.edu/tutoring or call our drop-in center during open business hours. Monday-Thursday 9am - 8pm on class days (254) 710-4135.

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

TOPIC OF THE WEEK

Pass by Value vs. Pass by Reference

Oftentimes when writing a function, we want to have the function return a variable in response to being called. 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. Thus, we need to understand the concept of ***Pass by Value*** vs. ***Pass by Reference***.

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

void doubleValue(int numByVal);

int main() {

    int number = 10;

    doubleValue(number);

    cout << number << endl;

    return 0;
}

void doubleValue(int numByVal) {
    numByVal *= 2;
}
```

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.

```
10

Process finished with exit code 0
```

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 '&' and 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. Thus, any modifications done to the variable used within the function, will be made to the function back in the main()

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 an integer so we specify int& to represent passing an int by

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

void doubleValue(int& numByRef);

int main() {

    int number = 10;

    doubleValue( &: number);

    cout << number << endl;

    return 0;
}

void doubleValue(int& numByRef) {
    numByRef *= 2;
}
```

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.

```
20

Process finished with exit code 0
```

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 instances done with “Returning from a function”, 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;
}
```

Thanks for checking out these weekly resources! Don't forget to check out our website for group tutoring times, video tutorials and lots of other course resources: https://baylor.edu/tutoring. The answers to the 'Check Your Learning' questions are below.

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