LW: Pass By Reference

Objectives

- To understand that:
  - argument passing is similar to initialization,
  - when a function is called, each formal argument is initialized by its corresponding actual argument, and
  - when a reference is provided as a formal parameter, the corresponding actual argument will be referred to by the formal parameter.

Labwork

- There are questions throughout this document that you will need to respond to for credit.
- You must compile and run this code on **build.tamu.edu**, executing the command `g++ LW_Pass-By-Ref.cpp` to compile, and then `./a.out` to run the resultant executable.
- Download the skeleton code for this lab from **https://drive.google.com/open?id=0B_ouNNuWgNZCb2k4aGQ5dkpRbnM**
- Upload the source code to your CSCE-drive.
- Using Putty (PC) or terminal (Mac), log-in to **build.tamu.edu**, and navigate to the directory containing `LW_Pass-By-Ref.cpp`.
- Inspect the source code; notice that the source code is identical to that in LW: Pass By Value, with the exception that the type of formal parameter to vint_half_sum is now a reference.
  a. How did the declaration of `vint_half_sum` change in order to pass the argument by reference instead of by value?

     The reference operator (&) was added to the input type in vint_half_sum. This indicates that the addresses of the values are to be input, not just the values themselves.

  b. How did the definition of `vint_half_sum` change in order to pass the argument by reference instead of by value?

     Also using &, it indicated that it was to take in the addresses of the values inside of v, which means that in this case, v is a reference to vint instead of a copy of it like int the previous activity.

c. With this in mind, understand that you can view the initialization of the formal parameter `vector<int>& v` in the function call to `vint_half_sum` on line 34 with `vint` as:
    i. vector<int>& v = vint;

- Compile and run the source code.
- Carefully, observe the output printed to the screen and then answer the following questions in the fields provided:
    a. Explain why the modification of `v` in `vint_half_sum` *does* mutate the actual argument `vint`:

As said above, v, for all intensive purposes, is vint. Any modifications to v are just referred back to vint for modification, as that is where the values are.

    b. What information included in the output produced by the calls to `vis::print` in `main` with `vint` and in `vint_half_sum` supports your response to 8a?

The addresses for both objects are the same, showing that they both reference the exact same set of values in memory.

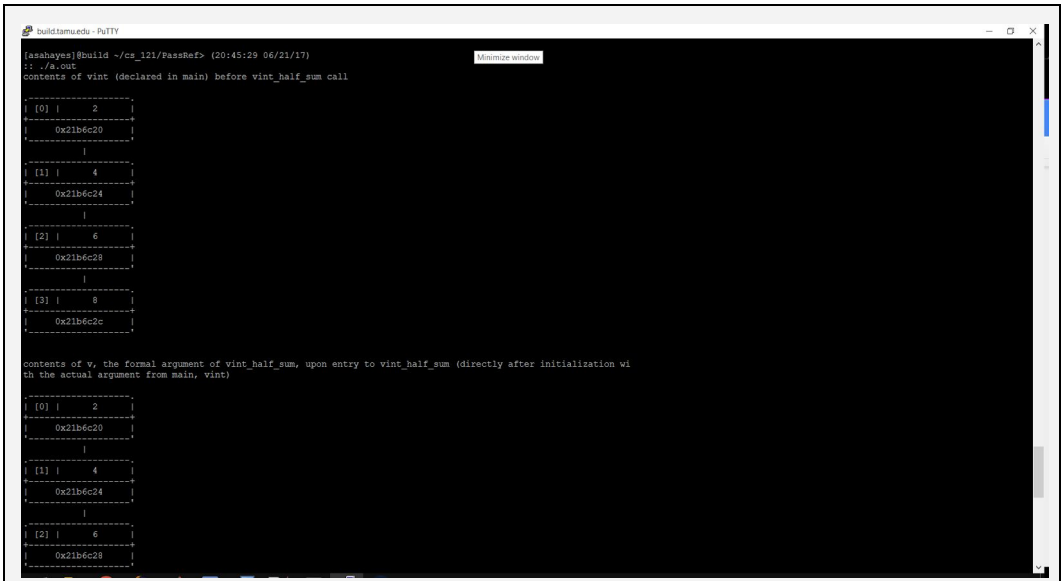
    c. What is the difference between pass-by-value and pass-by-reference? When might you use one over the other? Do you see any potential pitfalls in using either method?

Passby values just take in a copy of the data of the passed by object, while passby references take in the exact same data in the same memory address as the main object instead of creating a copy.

d. It is recommended that when passing a parameter by reference to a function you should denote in the function's identifier if it modifies the object in which a formal argument refers. Why do you think that this is a good idea? For instance, why might we decide to update the name of `vint_half_sum` to `half_elems_of_vint_ret_sum`?

> If you do not denote what is modifying the object and just have the references named the same as the original, it will be very difficult to determine which reference of the object is causing issues should any errors arise.

e. Capture the output written to the terminal window by this program in the form of a screenshot; if you cannot include everything, that's okay. Drag and drop or paste your screenshot into the box below:



Submission

- Save this completed labwork as a PDF [File -> Download As -> PDF Document (.pdf)] and submit to gradescope for grading.