LW: Pass By Value

Due by Fri, 15 Feb 2019 08:00:00-0600

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 variable is provided as a formal parameter, the corresponding actual argument's value will be copied into a local object of the called function.

Labwork

1. There are questions at the end of this document that you will respond to for credit.
2. Using Putty (PC) or terminal (Mac), log-in to **`compute.cse.tamu.edu`**
3. Create an empty directory for this labwork. In a terminal (e.g. putty) navigate to this directory.
4. Download the source code into your directory from: https://drive.google.com/open?id=0B_ouNNuWgNZCc1lYMHhSUERVWU0
5. Verify you copied the file with the following command:
   `ls`
6. Compile using the following command:
   `g++-7.2.0 -std=c++17 *.cpp`
7. Inspect the source code. Use your editor of choice for files on the CSE servers.
   a. You will encounter four calls to a function that you haven't seen before, `vis::print`.
      i. This (i.e., `vis::print`) is a function whose implementation you need not be concerned about: This function `vis::print` to output the elements of a vector for you. (Written by Michael Nowak.)
      ii. For instance, when calling `vis::print(v)`, where `v` was defined as a `vector<int>` initialized with `{1,11}` (i.e., `vector<int> v =`

`{1,11}`), the following output was generated:

```
.------------------.
| [0] |      1     |
+------------------+
|  0x7fe002404cb0  |
'------------------'
          |
.------------------.
| [1] |     11     |
+------------------+
|  0x7fe002404cb4  |
'------------------'
```

- The value of each element of the vector `v` is contained in its own box:

```
.------------------.
| [0] |      1     |
+------------------+
|  0x7fe002404cb0  |
'------------------'
          |
```

  - On the first line of the box, the integer contained between the brackets (here, `[0]`) is the index of that respective element in the vector `v`:

```
.-------
| [0] |
+-------
```

  - Also on the first line of the box is the value stored at that index in the vector (here, `1`):

```
-------------.
|      1     |
-------------+
```

  - The hexadecimal value shown on the second line of the box is the address in memory where that integer value is being stored (here, `v.at(0)`; this will likely be ***different*** each time you run your program):

```
+------------------+
|  0x7fa9b9500000  |
'------------------'
```

b. On line 27, you will see that a `vector<int>` named `vint`, is initialized with four elements, `{2, 4, 6, 8}`.
c. On line 30, you will see that we use the `vis::print` function to print `vint`'s contents after initialization.
d. On line 33, you will see the initialization of an object named `half_sum` of type `int` with the return value from `vint_half_sum(vint)`.

   i. Note: the value used to initialize `half_sum` will be computed by that function call.

e. Jumping down to the definition of `vint_half_sum` on line 51, you'll see that when `vint` is provided to `vint_half_sum` as an actual argument. The corresponding formal argument `vector<int> v` will be initialized as an element-wise copy of `vint`.
   i. Here, we can essentially view this process as the occurrence of an implicit `vector<int> v = vint;` statement that is evaluated directly before the execution of the function body of `vint_half_sum`..

f. Therefore, from line 33, the formal argument of the function is initialized with the actual argument, and then the execution of the program continues from function body of `vint_half_sum` on line 53.
   i. On line 54, prints `v` using `vis::print`.
   ii. From 55-62, calculates the "half sum" of `v`.
   iii. On line 65, use `vis::print` to display the elements of `v` directly before returning the half-sum from the function.
   iv. The return statement on line 66 returns a copy of the value stored in `sum`, which is used to initialize `half_sum`, the int object declared on line 33 in the main function.

g. On line 37, prints `vint` again using `vis::print` before returning from the program to the operating system.

8. Now that you understand what's going on, compile the code and run it.
9. 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 not mutate (i.e. change) the actual argument `vint:`

      Vint_half_sum is outside of the scope of main and its modification does not affect the value in the main function.

   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 values of the vector vint remain the same, but the memory addresses are different.

   c. 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 to the box below:

```
contents of vint (declared in main) before vint_half_sum call

.-------------------.
| [3]       8       |
+-------------------+
|     0x676c2c      |
+-------------------+
| [2]       6       |
+-------------------+
|     0x676c28      |
+-------------------+
| [1]       4       |
+-------------------+
|     0x676c24      |
+-------------------+
| [0]       2       |
+-------------------+
|     0x676c20      |
'-------------------'
    Size : 4
Capacity : 4


contents of v, the formal argument of vint_half_sum, upon entry to vint_half_sum
 (directly after initialization with the actual argument from main, vint)

.-------------------.
| [3]       8       |
+-------------------+
|     0x676c4c      |
+-------------------+
| [2]       6       |
+-------------------+
|     0x676c48      |
+-------------------+
| [1]       4       |
+-------------------+
|     0x676c44      |
+-------------------+
| [0]       2       |
+-------------------+
|     0x676c40      |
'-------------------'
    Size : 4
Capacity : 4
```

```
Half sum: 10


contents of vint (declared in main) after vint_half_sum call

.-------------------.
| [3]       8       |
+-------------------+
|     0x676c2c      |
+-------------------+
| [2]       6       |
+-------------------+
|     0x676c28      |
+-------------------+
| [1]       4       |
+-------------------+
|     0x676c24      |
+-------------------+
| [0]       2       |
+-------------------+
|     0x676c20      |
'-------------------'
    Size : 4
Capacity : 4

[dmimar382]@compute ~/CSCE121/Labs/Lab8> (14:33:41 02/14/19)
```

Submission

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