

Alexei Hazell

Professor Holtzbauer

Data Structures

February 22, 2018

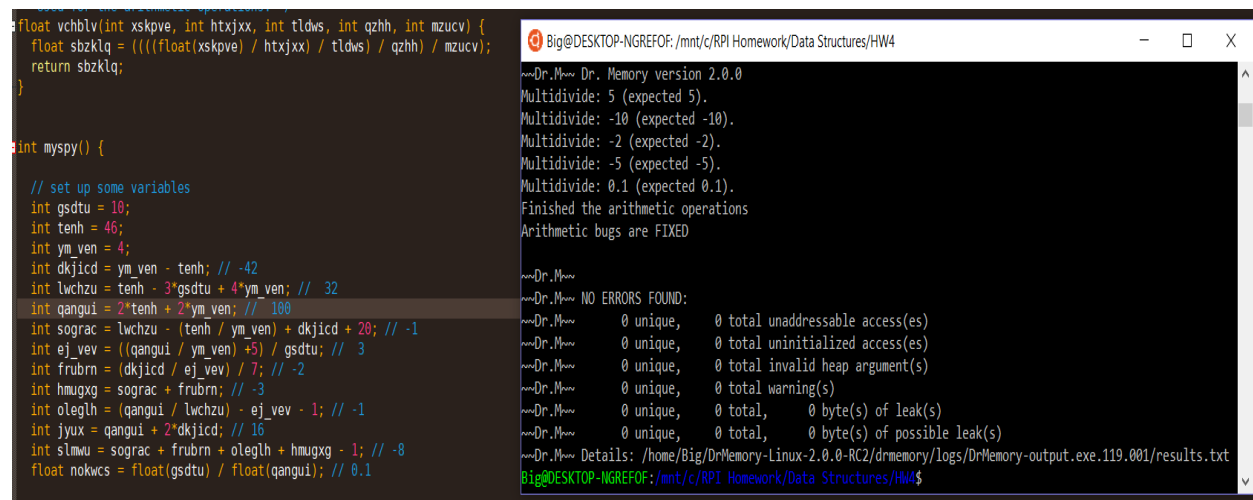
Detailed Bug Report

Operating System: Windows 7

Compiler: WSL

Bug #1 – Arithmetic Operations

One of the bugs that I encountered was a bug with the arithmetic operations. Some of the arithmetic was done incorrectly from line 332 to line 345. I discovered this error because the incorrect arithmetic caused some errors to occur in the assertions below in lines 352 to 383. Therefore, I needed to change the math so that the proper outputs. I included the changed and correct math as well as the output below.



The image shows a code editor on the left and a terminal window on the right. The code editor contains C code for a function `float vchblv` and a test function `int myspry`. The terminal window shows the output of the program, which includes several "Multidivide" assertions and a final "Arithmetic bugs are FIXED" message.

```
float vchblv(int xskpve, int htjxx, int tldws, int qzhh, int mzucv) {
    float sbzklq = (((float(xskpve) / htjxx) / tldws) / qzhh) / mzucv);
    return sbzklq;
}

int myspry() {
    // set up some variables
    int gsdtu = 10;
    int tenh = 46;
    int ym_ven = 4;
    int dkjcd = ym_ven - tenh; // -42
    int lwchzu = tenh - 3*gsdtu + 4*ym_ven; // 32
    int qangui = 2*tenh + 2*ym_ven; // 100
    int sograc = lwchzu - (tenh / ym_ven) + dkjcd + 20; // -1
    int ej_vev = ((qangui / ym_ven) + 5) / gsdtu; // 3
    int frubrn = (dkjcd / ej_vev) / 7; // -2
    int hmugxg = sograc + frubrn; // -3
    int oleglh = (qangui / lwchzu) - ej_vev - 1; // -1
    int jyux = qangui + 2*dkjcd; // 16
    int slmwu = sograc + frubrn + oleglh + hmugxg - 1; // -8
    float nokwcs = float(gsdtu) / float(qangui); // 0.1
}
```

```
Dr.Mem Dr. Memory version 2.0.0
Multidivide: 5 (expected 5).
Multidivide: -10 (expected -10).
Multidivide: -2 (expected -2).
Multidivide: -5 (expected -5).
Multidivide: 0.1 (expected 0.1).
Finished the arithmetic operations
Arithmetic bugs are FIXED

Dr.Mem
Dr.Mem NO ERRORS FOUND:
Dr.Mem 0 unique, 0 total unaddressable access(es)
Dr.Mem 0 unique, 0 total uninitialized access(es)
Dr.Mem 0 unique, 0 total invalid heap argument(s)
Dr.Mem 0 unique, 0 total warning(s)
Dr.Mem 0 unique, 0 total, 0 byte(s) of leak(s)
Dr.Mem 0 unique, 0 total, 0 byte(s) of possible leak(s)
Dr.Mem Details: /home/Big/DrMemory-Linux-2.0.0-RC2/drmemory/logs/DrMemory-output.exe.119.001/results.txt
Big@DESKTOP-NGREFOF:/mnt/c/RPI Homework/Data Structures/HW4$
```

Bug #2 - Logic

I encountered several problems with the logic gates in the file operations. First, I couldn't even start the operations for the section to run. I traced this problem to inside the function called "txhz". The problem was on line 252. The if statement was incorrect. Originally, it was `if(argc == 4)`. The variable `argc` is not equal to 4 so I changed it to `if(argc != 4)`. Another problem I encountered immediately after that was preventing me from even opening the file. Once again, on line 264, the if statement was incorrect. I read `if(mesne)` which meant when the actual file was found it would print out "That file could not be opened!". I changed the if statement to `if(not(mesne))` to correct it. On line 290, there was another logic error in the

assertion statement. The statement was `assert(mesne.gcount() != owsyz)`. This would mean that the entire file would not be read through and I expect the file to be completely read through. I changed the statement to `assert(mesne.gcount() == owsyz)` to make it what I expected.

I experienced some logic errors in the list operations as well. On line 117, the greater than sign should be flipped the other way into a less than sign. However, some other, non-logic based changes need to be made to make this one work. I needed to make `push_front` into `push_back` and I moved the for loop that was on lines 120 to 122 before the for loop on line 117. There was another issue on line 138 in the list operations section of code. There was an if statement that read `if(*fxaur % fcgvs_ != 0 || *fxaur % ii_n != 0)`. I want to remove something from the list if it is a multiple of 7 or 11. This would do the opposite of that so I fixed this by changing the not equals to signs to equal to signs. The if statement now reads `if(*fxaur % fcgvs_ == 0 || *fxaur % ii_n == 0)`.

There were definitely a few logic errors in the array operations section of the code. The first instance of a logic error in this section is on line 102. The if statement states `if((zwlh = 0))`. There is another logic error exactly like this on line 108. I just added another equal sign to make them the Boolean operator.

Bug #3 – Memory

I did not have too many memory errors but I did come across a few big ones. The first one I saw was between lines 95 and 126. First, there was a double pointer variable that was not set to anything on the heap. I fixed that quickly. After that, I kept getting memory leaks. I quickly discovered that I was getting those memory leaks because I was not deleting the pointer variable. Another memory issue that I had was in my function that adds my vector together. This one in particular puzzled me. My vector was adding incorrectly and I couldn't figure out why. The for loop that I fixed was in perfect shape. I then realized that in order to change the contents of the vector, I would have to pass it by reference. I preceded to put the '&' in the function prototypes and in the actual function. I also came across a memory leak in the main function. There was a pointer that was never deleted and I changed that.

Bug #4 – The Erase Function

I had some minor problems with the two erase function in the code. There is one at line 139 and one on line 185. In the loop from lines 137 to 141, I have to remove every number that is a multiple of either 7 or 11. The erase function here does that. However, because, after using the erase function on the list, it is not set back to the iterator. Therefore, I did just that. The second erase function on line 185 clearly has something wrong. The iterator in the erase function has '++' in front of it. The iterator goes through the list and would want to remove the fruit from the line it's on, not the next line. I deleted the '++' and my error disappeared.

Bug #5 – Out of Order

There were a couple times in which there was code that was simply just out of order. On line 273, the char pointer is initialized out of order. The way it was, it would just be making an array of size zero. This would cause errors many errors in code further on. To fix it, I moved this line, `char* vxzql = new char[owsyz];`, after `owsyz = mesne.tellg();`. Another set of code that was out of order was found on lines 120 to 122. This for loop needs to be put in front of the one that proceeds it. It also needs to have its `push_front` changed to `push_back` but by having the for loops switched the list will be in the correct order.

Bug #6 – No Initialization When Necessary

Some of these variables are not initialized when they should be. On line 205, `int xl_h`, `xl_h` is not initialized to any number. There is another instance of this on line 543 as well. These variables need to be initialized to a number, preferably 0, because later in the code they have '++' after or before them. In order for '++' to work it needs to be able to add 1 to a value that exists.