

Lab2 Question 4 Debugging Report

Fixing Syntactic Errors:

```
41     array[i].intData = temp.intData;
42     array[i].charData = temp.charData;
43
44     array[i + 1].intData = array[i].intData;
45     array[i + 1].charData = array[i].charData;
46
47     done = 0;
48 }
49 }
```

error: expected ';' before 'array'

Fix: Add a semicolon at the end of line 44.

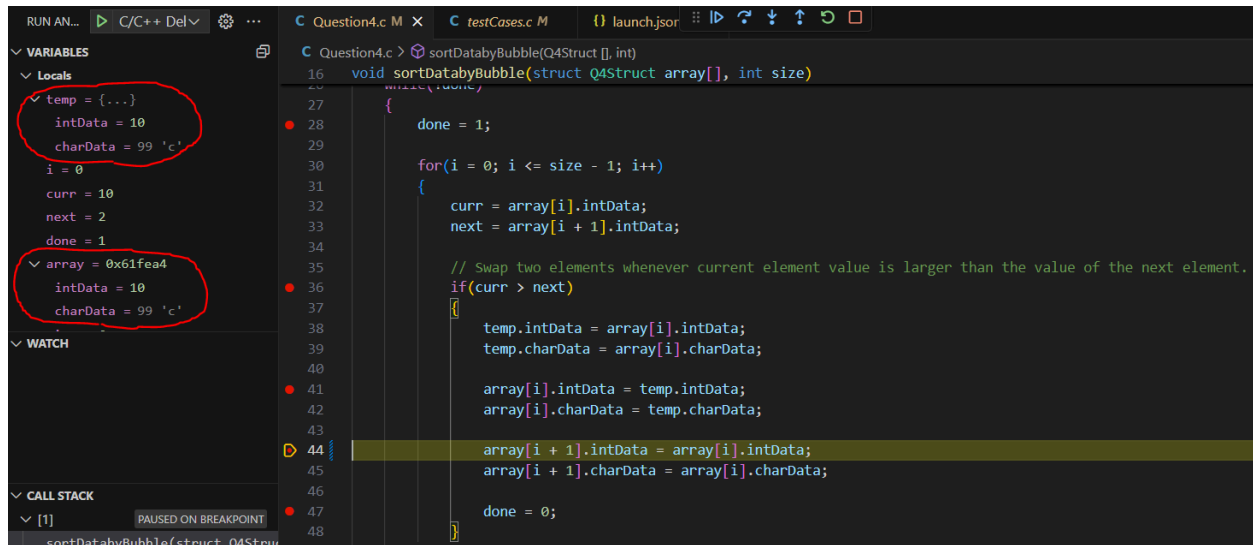
Fixing Semantic Bugs:

Bug #1: Incorrect Swapping Logic

The screenshot shows a C++ IDE with a bubble sort implementation. The left sidebar displays the 'LOCALS' and 'WATCH' panels. In the 'LOCALS' panel, the 'temp' variable is highlighted with a red circle, showing its current values: 'intData = 10' and 'charData = 99'. Below it, the 'array' variable is also highlighted with a red circle, showing 'intData = 10' and 'charData = 99'. The main editor shows the code for 'sortDataByBubble'. Lines 38 and 39 are highlighted in green, indicating they have been executed. The code assigns values from 'array[i]' to 'temp' and then from 'array[i+1]' to 'array[i]'. The 'CALL STACK' panel at the bottom shows the current function call.

As shown above, lines 38 and 39 have been executed. $i = 0$, meaning this is the very first pass in the for loop.

Lines 38 and 39 assign the first int and char elements in `array[i]` to a `temp[i]` variable, meant to store the values temporarily for swapping.



I've now clicked continue, which executes lines 41 and 42 and stops at the next breakpoint on line 44. Lines 41 and 42 assign the int and char elements in temp[i] to array[i]. However, this is **not correct**. The elements in array[i] were assigned to temp[i] so that array[i + 1] could be assigned to array[i], and then the values in temp[i] could be assigned to array[i + 1]. In other words, this was supposed to happen:

```
temp[i] = array[i]
```

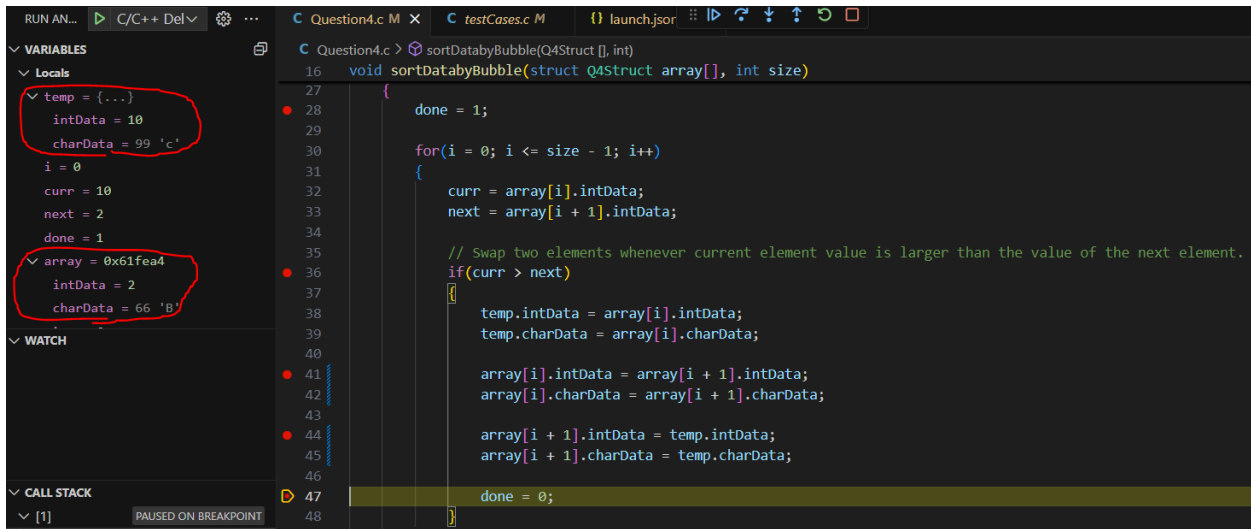
```
array[i] = array[i + 1]
```

```
array[i + 1] = temp[i]
```

However, that did not happen. This happened:

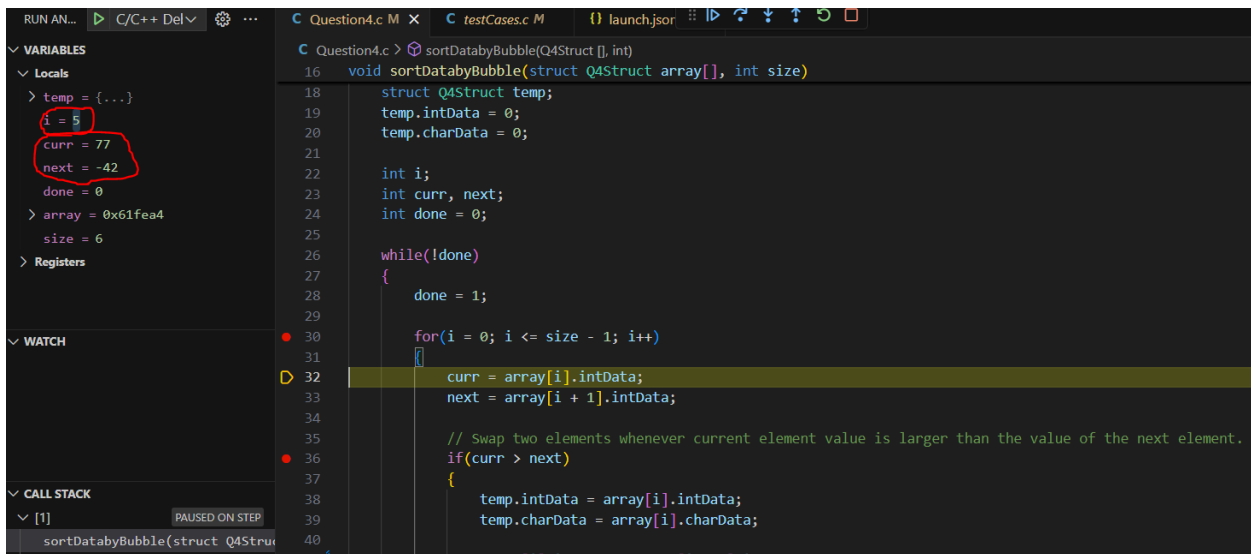
temp[i] = array[i] and then array[i] = temp[i], which achieves nothing, and then array[i + 1] = array[i]. So after all that, temp[i] = array[i] = array[i + 1]. This can be seen in the above image circled in red. The temp values and array values are all the same.

To fix this, we rewrite the code:



As shown above, the swapping logic has been rewritten. Running the debugger again, after 1 pass of the for loop, the elements in `array[i]` and `array[i + 1]` have been successfully swapped as `temp[i] = array[i + 1] = 10` and `'c'` and `array[i] = 2` and `'B'`.

Bug #2: For loop out of bounds



As shown in the screenshot above, `i` has just been incremented to 5. `i = size - 1`, so it enters the for loop. `curr` and `next` are still their previous values from `i = 4`.

```
struct Q4Struct input[]={10, 'c'}, {2, 'B'}, {-5, 'k'}, {12, 'z'}, {77, 'a'}, {-42, '?'}};
```

As you can see, those are the last `i` and `i + 1`th int values in the array.

```
void sortDataByBubble(struct Q4Struct array[], int size)
{
    struct Q4Struct temp;
    temp.intData = 0;
    temp.charData = 0;

    int i;
    int curr, next;
    int done = 0;

    while(!done)
    {
        done = 1;

        for(i = 0; i <= size - 1; i++)
        {
            curr = array[i].intData;
            next = array[i + 1].intData;

            // Swap two elements whenever current element value is larger than the value of the next element.
            if(curr > next)
            {
                temp.intData = array[i].intData;
                temp.charData = array[i].charData;
```

Next, lines 32 and 33 are executed. Next is assigned the $i + 1$ th element of the array, which is $5 + 1 = 6$. So, it attempts to retrieve `array[6]`, the 7th element of the array, which doesn't exist. A junk value of 6 is then assigned to next, causing a Segmentation error.

To fix this, we simply change $i \leq \text{size} - 1$ to $i < \text{size} - 1$. This ensures that the for loop doesn't attempt to retrieve a value from the array that doesn't exist.

```
void sortDataByBubble(struct Q4Struct array[], int size)
{
    struct Q4Struct temp;
    temp.intData = 0;
    temp.charData = 0;

    int i;
    int curr, next;
    int done = 0;

    while(!done)
    {
        done = 1;

        for(i = 0; i < size - 1; i++)
        {
            curr = array[i].intData;
            next = array[i + 1].intData;

            // Swap two elements whenever current element value is larger than the value of the next element.
            if(curr > next)
            {
                temp.intData = array[i].intData;
                temp.charData = array[i].charData;
```

This fixes the segmentation fault issue as the final element retrieved in `array[]` will be `array[5]`, the 6th and final element of the array.