5.10 Debugging example: Reversing a vector

A common vector modification is to reverse a vector's elements. One way to accomplish this goal is to perform a series of swaps. For example, starting with a vector of numbers 10 20 30 40 50 60 70 80, we could first swap the first item with the last item, yielding 80 20 30 40 50 60 70 10. We could next swap the second item with the second-to-last item, yielding 80 70 30 40 50 60 20 10. The next swap would yield 80 70 60 40 50 30 20 10, and the last would yield 80 70 60 50 40 30 20 10.

With this basic idea of how to reverse a vector, we can attempt to write a program to carry out such reversal. Below we develop such a program but we make common mistakes along the way, to aid learning from examples of what not to do.

A first attempt to write a program that reverses a vector appears below.

Figure 5.10.1: First program attempt to reverse vector: Aborts due to invalid access of vector element.

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

int main() {
   const int NUM_ELEMENTS = 8;         // Number
of elements
   vector<int> revVctr(NUM_ELEMENTS); // User
values
   unsigned int i;                     // Loop
index

   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < revVctr.size(); ++i) {
      cout << "Value: ";
      cin >> revVctr.at(i);
   }

   // Reverse
   for (i = 0; i < revVctr.size(); ++i) {
      revVctr.at(i) = revVctr.at(revVctr.size() -
i); // Swap
   }

   // Print values
   cout << endl << "New values: ";
   for (i = 0; i < revVctr.size(); ++i) {
      cout << " " << revVctr.at(i);
   }
   cout << endl;

   return 0;
}
```

```
Enter 8 integer values...
Value: 10
Value: 20
Value: 30
Value: 40
Value: 50
Value: 60
Value: 70
Value: 80
libc++abi.dylib: terminating with
uncaught exception
of type std::out_of_range: vector
```

Feedback?

Something went wrong: The program aborted (exited abnormally). The reported message indicates an "out of range" problem related to a vector, meaning the program tried to access a vector element that doesn't exist. Let's try to find the code that caused the problem.

The first and third for loops are fairly standard, so let's initially focus attention on the middle for loop that does the reversing. The swap statement inside that loop is `revVctr.at(i) = revVctr.at(revVctr.size() - i)`. When i is 0, the statement will execute `revVctr.at(0) = revVctr.at(8)`. However, revVctr has size 8 and thus valid indices are 0..7. revVctr.at(8) does not exist. The program should actually swap elements 0 and 7, then 1 and 6, etc. Thus, let's change the right-side index to `revVctr.size() - 1 - i`. The revised program is shown below.

Figure 5.10.2: Revised vector reversing program: Doesn't abort, but still a problem.

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

int main() {
   const int NUM_ELEMENTS = 8;         // Number of
elements
   vector<int> revVctr(NUM_ELEMENTS); // User values
   unsigned int i;                    // Loop index

   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < revVctr.size(); ++i) {
      cout << "Value: ";
      cin >> revVctr.at(i);
   }

   // Reverse
   for (i = 0; i < revVctr.size(); ++i) {
      revVctr.at(i) = revVctr.at(revVctr.size() - 1 - i);
// Swap
   }

   // Print values
   cout << endl << "New values: ";
   for (i = 0; i < revVctr.size(); ++i) {
      cout << " " << revVctr.at(i);
   }
   cout << endl;

   return 0;
}
```

```
Enter 8 integer values...
Value: 10
Value: 20
Value: 30
Value: 40
Value: 50
Value: 60
Value: 70
Value: 80

New values:  80 70 60 50 50 60
70 80
```

Feedback?

The program didn't abort this time, but the last four elements are wrong. To determine what went wrong, we can manually (i.e., on paper) trace the loop's execution.

- i is 0: revVctr.at(0) = revVctr.at(7). Vector now: 80 20 30 40 50 60 70 80.
- i is 1: revVctr.at(1) = revVctr.at(6). Vector now: 80 70 30 40 50 60 70 80.
- i is 2: revVctr.at(2) = revVctr.at(5). Vector now: 80 70 60 40 50 60 70 80.
- i is 3: revVctr.at(3) = revVctr.at(4). Vector now: 80 70 60 50 50 60 70 80.
- i is 4: revVctr.at(4) = revVctr.at(3). Vector now: 80 70 60 50 50 60 70 80. *Uh-oh, where did 40 go?*

We failed to actually swap the vector elements, instead the code just copies values in one direction. We need to add code to properly swap. We add a variable tmpValue to temporarily hold `revVctr.at(revVctr.size() - 1 - i)` so we don't lose that element's value.

Figure 5.10.3: Revised vector reversing program with proper swap: Output isn't reversed.

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

int main() {
   const int NUM_ELEMENTS = 8;         // Number of
elements
   vector<int> revVctr(NUM_ELEMENTS); // User values
   unsigned int i;                     // Loop index
   int tmpValue;                       // Placeholder

   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < revVctr.size(); ++i) {
      cout << "Value: ";
      cin >> revVctr.at(i);
   }

   // Reverse
   for (i = 0; i < revVctr.size(); ++i) {
      tmpValue = revVctr.at(i);  // These 3 statements
swap
      revVctr.at(i) = revVctr.at(revVctr.size() - 1 - i);
      revVctr.at(revVctr.size() - 1 - i) = tmpValue;
   }

   // Print values
   cout << endl << "New values: ";
   for (i = 0; i < revVctr.size(); ++i) {
      cout << " " << revVctr.at(i);
   }
   cout << endl;

   return 0;
}
```

```
Enter 8 integer values...
Value: 10
Value: 20
Value: 30
Value: 40
Value: 50
Value: 60
Value: 70
Value: 80

New values:  10 20 30 40 50 60
70 80
```

Feedback?

The new values are not reversed. Again, let's manually trace the loop iterations.

- i is 0: revVctr.at(0) = revVctr.at(7). Vector now: 80 20 30 40 50 60 70 10.
- i is 1: revVctr.at(1) = revVctr.at(6). Vector now: 80 70 30 40 50 60 20 10.
- i is 2: revVctr.at(2) = revVctr.at(5). Vector now: 80 70 60 40 50 30 20 10.
- i is 3: revVctr.at(3) = revVctr.at(4). Vector now: 80 70 60 50 40 30 20 10. *Looks reversed.*
- i is 4: revVctr.at(4) = revVctr.at(3). Vector now: 80 70 60 40 50 30 20 10. *Why are we still swapping?*

Tracing makes clear that the for loop should not iterate over the entire vector. The reversal is completed halfway through the iterations. The solution is to set the loop expression to `i < (revVctr.size() / 2)`.

Figure 5.10.4: Vector reversal program with correct output.

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

int main() {
   const int NUM_ELEMENTS = 8;         // Number of
elements
   vector<int> revVctr(NUM_ELEMENTS); // User values
   unsigned int i;                     // Loop index
   int tmpValue;                       // Placeholder

   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < revVctr.size(); ++i) {
      cout << "Value: ";
      cin >> revVctr.at(i);
   }

   // Reverse
   for (i = 0; i < (revVctr.size() / 2); ++i) {
      tmpValue = revVctr.at(i);  // These 3 statements
swap
      revVctr.at(i) = revVctr.at(revVctr.size() - 1 - i);
      revVctr.at(revVctr.size() - 1 - i) = tmpValue;
   }

   // Print values
   cout << endl << "New values: ";
   for (i = 0; i < revVctr.size(); ++i) {
      cout << " " << revVctr.at(i);
   }
   cout << endl;

   return 0;
}
```

```
Enter 8 integer values...
Value: 10
Value: 20
Value: 30
Value: 40
Value: 50
Value: 60
Value: 70
Value: 80

New values:  80 70 60 50 40 30
20 10
```

Feedback?

We should ensure the program works if the number of elements is odd rather than even. Suppose the vector has 5 elements (0-4) with values 10 20 30 40 50. revVctr.size() / 2 would be 5 / 2 = 2, meaning the loop expression would be i < 2. The iteration when i is 0 would swap elements 0 and 4 (5-1-0), yielding 50 20 30 40 10. The iteration for i=1 would swap elements 1 and 3, yielding 50 40 30 20 10. The loop would then not execute again because i is 2. So the results are correct for an odd number of elements, because the middle element will just not move.

The mistakes made above are each very common when dealing with loops and vectors, especially for beginning programmers. An incorrect (in this case out-of-range) index, an incorrect swap, and an incorrect loop expression. The lesson is that loops and vectors require attention to detail, greatly aided by manually executing the loop to determine what is happening on each iteration. Ideally, a programmer will take more care when writing the original program, but the above mistakes are quite common.

PARTICIPATION ACTIVITY 5.10.1: Find the error in the vector reversal code.

1)
```
for (i = 0; i < prices.size(); ++i) {
   tmp = prices.at(i);
   prices.at(i) =
prices.at(prices.size() - 1 - i);

prices.at(prices.size() - 1 - i) = tmp;
}
```

2)
```
for (i = 0; i < (prices.size() / 2); ++i) {
   tmp = prices.at(i);
   prices.at(i) =
prices.at(prices.size() - i);
   prices.at(prices.size() - i - 1) = tmp;
}
```

3)
```
for (i = 0; i < (prices.size() / 2); ++i) {
   tmp = prices.at(i);
   prices.at(prices.size() - i - 1) = tmp;

prices.at(i) = prices.at(prices.size() - 1 - i);
}
```

Feedback?