

## 5.8 Loop-modifying or copying/comparing vectors

### Modifying vector elements

A program may need to modify elements while iterating through a vector. The program below uses a loop to convert any negative vector element value to 0.

Figure 5.8.1: Modifying a vector during iteration example: Converting negatives to 0.

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

int main() {
    const int NUM_ELEMENTS = 5;           // Number of elements
    vector<int> userVals(NUM_ELEMENTS);    // User values
    unsigned int i;                       // Loop index

    // Prompt user to populate vector
    cout << "Enter " << NUM_ELEMENTS << " integer values..." << endl;
    for (i = 0; i < userVals.size(); ++i) {
        cout << "Value: ";
        cin >> userVals.at(i);
    }

    // Convert negatives to 0
    for (i = 0; i < userVals.size(); ++i) {
        if (userVals.at(i) < 0) {
            userVals.at(i) = 0;
        }
    }

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

    return 0;
}
```

```
Enter 5 integer values...
Value: 67
Value: -5
Value: -99
Value: 4
Value: 22
New values: 67 0 0 4 22
```

[Feedback?](#)

What is the resulting vector contents, assuming each question starts with a vector of size 4 having contents -55, -1, 0, 9?

1) 

```
for (i = 0; i < 4; ++i) {  
    itemsList.at(i) = i;  
}
```

- ☐ -54, 0, 1, 10
- ☐ 0, 1, 2, 3
- ☐ 1, 2, 3, 4

2) 

```
for (i = 0; i < 4; ++i) {  
    if (itemsList.at(i) < 0) {  
        itemsList.at(i) =  
            itemsList.at(i) * -1;  
    }  
}
```

- ☐ -55, -1, 0, -9
- ☐ 55, 1, 0, -9
- ☐ 55, 1, 0, 9

3) 

```
for (i = 0; i < 4; ++i) {  
    itemsList.at(i) =  
        itemsList.at(i+1);  
}
```

- ☐ -1, 0, 9, 0
- ☐ 0, -55, -1, 0
- ☐ Error (program aborts)

4) 

```
for (i = 0; i < 3; ++i) {  
    itemsList.at(i) =  
        itemsList.at(i+1);  
}
```

- ☐ -1, 0, 9, 9
- ☐ Error (program aborts)
- ☐ -1, 0, 9, 0

5) 

```
for (i = 0; i < 3; ++i) {  
    itemsList.at(i+1) =  
        itemsList.at(i);  
}
```

- ☐ -55, -55, -55, -55
- ☐ 0, -55, -1, 0
- ☐ Error (program aborts)

## zyDE 5.8.1: Modifying a vector during iteration example: Doubling element values.

Complete the following program to double each number in the vector.

[Load default template...](#)

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     const int NUM_ELEMENTS = 5;           // !
7     vector<int> userVals(NUM_ELEMENTS);    // !
8     unsigned int i;                       // !
9
10    // Prompt user to populate vector
11    cout << "Enter " << NUM_ELEMENTS << " i: ";
12    for (i = 0; i < userVals.size(); ++i) {
13        cout << "Value: " << endl;
14        cin >> userVals.at(i);
15    }
16
17    // Convert negatives to 0
18    for (i = 0; i < userVals.size(); ++i) {
19        if (userVals.at(i) < 0) {
20            userVals.at(i) = 0;
21        }

```

67 -5 -99 4 22

Run

[Feedback?](#)

## Element by element vector copy

In C++, the = operator conveniently performs an element-by-element copy of a vector, called a **vector copy operation**. The operation `vectorB = vectorA` resizes `vectorB` to `vectorA`'s size, appending or deleting elements as needed. `vectorB` commonly has a size of 0 before the operation.

Figure 5.8.2: Using = to copy a vector: Original and sale prices.

|                  |             |
|------------------|-------------|
| Original prices: | 10 20 30 40 |
| Sale prices:     | 10 20 27 35 |

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

int main() {
    const int    NUM_ELEMENTS = 4;           // Number of elements
    vector<int> origPrices(NUM_ELEMENTS);    // Original prices
    vector<int> salePrices(NUM_ELEMENTS);    // Sale prices
    unsigned int i;                          // Loop index

    // Assign original prices
    origPrices.at(0) = 10;
    origPrices.at(1) = 20;
    origPrices.at(2) = 30;
    origPrices.at(3) = 40;

    // Copy original prices to sales prices
    salePrices = origPrices;

    // Update salePrices. Note: does not affect origPrices
    salePrices.at(2) = 27;
    salePrices.at(3) = 35;

    // Output original and sale prices
    cout << "Original prices: ";
    for (i = 0; i < origPrices.size(); ++i) {
        cout << " " << origPrices.at(i);
    }
    cout << endl;

    cout << "Sale prices:      ";
    for (i = 0; i < salePrices.size(); ++i) {
        cout << " " << salePrices.at(i);
    }
    cout << endl;

    return 0;
}
```

[Feedback?](#)**PARTICIPATION  
ACTIVITY**

## 5.8.2: Vector copy operation.

Assume vectors have been declared as follows and have been initialized as indicated in the comments:

```
vector<int> userVals(4); // {44, 55, 66, 77}
vector<int> newVals;    // No elements yet
```

- 1) What is newVals after: `newVals = userVals;`

Type answer as: 10, 20, 30, 40

If appropriate type: Error

**Check**[Show answer](#)

2) What is newVals after:

```
newVals = userVals;  
userVals.at(0) = 33;
```

Type answer as: 10, 20, 30, 40

If appropriate type: Error

[Check](#)[Show answer](#)

3) Given: vector<int> otherVals(9).

What size is newVals after:

```
newVals = userVals;  
...
```

```
newVals = otherVals;
```

If appropriate type: Error

[Check](#)[Show answer](#)[Feedback?](#)

## Element by element vector comparison

In C++, the == operator conveniently compares vectors element-by-element, called a **vector equality operation**, with vectorA == vectorB evaluating to true if the vectors are the same size AND each element pair is equal.

### PARTICIPATION ACTIVITY

#### 5.8.3: Vector copying.

Assume vectors have been declared as follows and have been initialized as indicated in the comments:

```
vector<int> vectorX(2); // {3,4}  
vector<int> vectorY(5); // {3,4,0,7,8}  
vector<int> vectorZ(5); // {3,4,0,6,8}
```

1) (vectorX == vectorY) will evaluate to:

- ☐ True  
☐ False

2) Given: `vectorX = vectorY;` (`vectorX == vectorY`) will evaluate to:

- ☐ True  
☐ False

3) (`vectorZ == vectorY`) will evaluate to:

- ☐ True  
☐ False

4) (`vectorZ.size() == vectorY.size()`) will evaluate to:

- ☐ True  
☐ False

[Feedback?](#)

**CHALLENGE  
ACTIVITY**

5.8.1: Decrement vector elements.



Write a loop that subtracts 1 from each element in `lowerScores` if the original element was greater than 0, and otherwise just assigns the element with 0. Ex: `lowerScores = {5, 0, 2, -3}` becomes `{4, 0, 1, 0}`.

```
3 using namespace std;
4
5 int main() {
6     const int SCORES_SIZE = 4;
7     vector<int> lowerScores(SCORES_SIZE);
8     unsigned int i;
9
10    for (i = 0; i < lowerScores.size(); ++i) {
11        cin >> lowerScores.at(i);
12    }
13
14    /* Your solution goes here */
15    for (i = 0; i < lowerScores.size(); ++i) {
16        if (lowerScores.at(i) > 0)
17        {
18            lowerScores.at(i)--;
19        }else{
20            lowerScores.at(i) = 0;
21        }
22    }
23
24    for (i = 0; i < lowerScores.size(); ++i) {
```

**Run**

✓ All tests passed

✓ Testing with inputs: 5 0 2 -3

Your output 4 0 1 0

✓ Testing with inputs: 100 80 60 40

Your output 99 79 59 39

✓ Testing with inputs: -6 21 0 401

Your output 0 20 0 400

[Feedback?](#)**CHALLENGE  
ACTIVITY**

## 5.8.2: Copy and modify vector elements.



Write a loop that sets newScores to oldScores shifted once left, with element 0 copied to the end. Ex: If oldScores = {10, 20, 30, 40}, then newScores = {20, 30, 40, 10}.

Note: These activities may test code with different test values. This activity will perform two tests, both with a 4-element array. See ["How to Use zyBooks"](#).

Also note: If the submitted code tries to access an invalid array element, such as newScores[9] for a 4-element array, the test may generate strange results. Or the test may crash and report "Program end never reached", in which case the system doesn't print the test case that caused the reported message.

```

6   const int SCORES_SIZE = 4;
7   vector<int> oldScores(SCORES_SIZE);
8   vector<int> newScores(SCORES_SIZE);
9   unsigned int i;
10
11   for (i = 0; i < oldScores.size(); ++i) {
12       cin >> oldScores.at(i);
13   }
14
15   /* Your solution goes here */
16   for (i = 0; i < oldScores.size()-1; ++i) {
17       newScores.at(i) = oldScores.at(i+1);
18   }
19   newScores.at(oldScores.size()-1) = oldScores.at(0);
20
21   for (i = 0; i < newScores.size(); ++i) {
22       cout << newScores.at(i) << " ";
23   }
24   cout << endl;
25
26   return 0;

```

**Run**

✓ All tests passed

✓ Testing with inputs: 10 20 30 40

Your output 20 30 40 10

✓ Testing with inputs: 199 80 125 60

Your output 80 125 60 199

[Feedback?](#)**CHALLENGE  
ACTIVITY**

## 5.8.3: Modify vector elements using other elements.



Write a loop that sets each vector element to the sum of itself and the next element, except for the last element which stays the same. Be careful not to index beyond the last element. Ex:

**Initial scores:** 10, 20, 30, 40

**Scores after the loop:** 30, 50, 70, 40

The first element is 30 or  $10 + 20$ , the second element is 50 or  $20 + 30$ , and the third element is 70 or  $30 + 40$ . The last element remains the same.

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6      const int SCORES_SIZE = 4;
7      vector<int> bonusScores(SCORES_SIZE);
8      unsigned int i;
9
10     for (i = 0; i < bonusScores.size(); ++i) {
11         cin >> bonusScores.at(i);
12     }
13
14     /* Your solution goes here */
15     for (i = 0; i < bonusScores.size()-1; ++i) {
16         bonusScores.at(i) = bonusScores.at(i) + bonusScores.at(i+1);
17     }
18
19     for (i = 0; i < bonusScores.size(); ++i) {
20         cout << bonusScores.at(i) << " ";
21     }

```

**Run**



**Run**

✓ All tests passed

✓ Testing with inputs: 10 20 30 40

Your output 30 50 70 40

✓ Testing with inputs: 199 299 399 499

Your output 498 698 898 499

✓ Testing with inputs: -100 -200 -300 -400

Your output -300 -500 -700 -400

[Feedback?](#)**CHALLENGE  
ACTIVITY**

## 5.8.4: Modify a vector's elements.



Subtract 4 to any element's value that is greater than maxVal. Ex: If maxVal = 10, then dataPoints = {2, 12, 9, 20} becomes {2, 8, 9, 16}.

```
4
5 int main() {
6     int maxVal;
7     const int NUM_POINTS = 4;
8     vector<int> dataPoints(NUM_POINTS);
9     unsigned int i;
10
11     cin >> maxVal;
12
13     for (i = 0; i < dataPoints.size(); ++i) {
14         cin >> dataPoints.at(i);
15     }
16
17     /* Your solution goes here */
18     for (i = 0; i < dataPoints.size(); ++i) {
19         if (dataPoints.at(i) > maxVal)
20         {
21             dataPoints.at(i) = dataPoints.at(i)-4;
22         }
23     }
24 }
```

**Run**

✓ All tests passed

✓ Testing with inputs: 10 2 12 9 20

Your output

2 8 9 16

✓ Testing with inputs: 4 1 2 3 4

Your output

1 2 3 4

[Feedback?](#)**CHALLENGE  
ACTIVITY**

## 5.8.5: Comparing and copying vectors.



If the vector `oldData` is the same as the vector `newData`, print "Data matches!" ended with a newline. Otherwise, assign `oldData` with `newData`. Ex: If `oldData` = {10, 12, 18, 16} and `newData` = {25, 27, 29, 23}, then `oldData` becomes {25, 27, 29, 23}.

```

13     }
14
15     for (i = 0; i < newData.size(); ++i) {
16         cin >> newData.at(i);
17     }
18
19     /* Your solution goes here */
20     /*for (i = 0; i < newData.size(); ++i) {
21         if(oldData.at(i)!=newData.at(i)){
22             oldData = newData;
23         }else{
24             cout<< "Data matches!"<< endl;
25         }
26     }*/
27     if(oldData == newData){
28
29         cout<< "Data matches!"<< endl;
30     }else{
31         oldData = newData;
32     }
33
34     for (i = 0; i < oldData.size(); ++i) {

```

**Run**

✓ All tests passed

✓ Testing with inputs: 10 12 18 16 25 27 29 23

Your output

25 27 29 23

✓ Testing with inputs: 22 33 44 0 7 3 6 5

Your output

7 3 6 5

✓ Testing with inputs: 0 5 0 5 0 5 0 5

Your output

Data matches!  
0 5 0 5

[Feedback?](#)