5.4 Iterating through vectors

Iterating through vectors using loops

Iterating through vectors using loops is commonplace and is an important programming skill to master. Because vector indices are numbered 0 to N - 1 rather than 1 to N, programmers commonly use this for loop structure:

Figure 5.4.1: Common for loop structure for iterating through a vector.

```
// Iterating through myVector
for (i = 0; i < myVector.size(); ++i) {
   // Loop body accessing myVector.at(i)
}
```

Feedback?

Note that index variable i is initialized to 0, and the loop expression is i < myVector.size() rather than i <= myVector.size(). If myVector.size() were 5, the loop's iterations would set i to 0, 1, 2, 3, and 4, for a total of 5 iterations. The benefit of the loop structure is that each vector element is accessed as myVector.at(i) rather than the more complex myVector.at(i - 1).

PARTICIPATION ACTIVITY 5.4.1: Iterating through a vector.

1) Complete the code to print all items for the given vector, using the above common loop structure.

```
vector<int> daysList(365);

for (i = 0;
                        ; ++i) {
   cout << daysList.at(i) <<
endl;
}
```

Check Show answer

2) Given that this loop iterates over all items of the vector, how many items are in the vector?

```
for (i = 0; i < 99; ++i) {
   cout << someVector.at(i) << endl;
}
```

Check Show answer

Feedback?

Determining a quantity about a vector's items

Iterating through a vector for various purposes is an important programming skill to master. Programs commonly iterate through vectors to determine some quantity about the vector's items. The example below computes the sum of a vector's element values.

Figure 5.4.2: Iterating through a vector example: Program that finds the sum of a vector's elements.

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

int main() {
   const int NUM_ELEMENTS = 8;          // Number of elements in
vector
   vector<int> userVals(NUM_ELEMENTS); // User values
   unsigned int i;                      // Loop index
   int sumVal;                          // For computing sum

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

   // Determine sum
   sumVal = 0;
   for (i = 0; i < userVals.size(); ++i) {
      sumVal = sumVal + userVals.at(i);
   }
   cout << "Sum: " << sumVal << endl;

   return 0;
}
```

```
Enter 8 integer
values...
Value: 3
Value: 5
Value: 234
Value: 346
Value: 234
Value: 73
Value: 26
Value: -1
Sum: 920
```

Feedback?

Finding the maximum value in a vector

The program below determines the maximum value in a user-entered list. If the user enters numbers 7, -9, 55, 44, 20, -400, 0, 2, then the program will output "max: 55". The program uses the variable maxVal to store the largest value seen thus far as the program iterates through the vector. During each iteration, if the vector's current element value is larger than the max seen thus far, the program assigns maxVal with the current vector element.

Before entering the loop, maxVal must be initialized to some value because maxVal will be compared with each vector element's value. A logical error would be to initialize maxVal to 0, because 0 is not in fact the largest value seen so far, and would result in incorrect output (of 0) if the user entered all negative numbers. Instead, the program peeks at a vector element (in this case the first element, though any element could be used) and initializes maxVal with that element's value.

Figure 5.4.3: Iterating through a vector example: Program that finds the max item.

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

int main() {
   const int NUM_VALS = 8;          // Number of elements in vector
   vector<int> userVals(NUM_VALS); // User values
   unsigned int i;                  // Loop index
   int maxVal;                      // Computed max

   cout << "Enter " << NUM_VALS << " integer numbers..." << endl;
   for (i = 0; i < userVals.size(); ++i) {
      cout << "Value: ";
      cin >> userVals.at(i);
      cout << endl;
   }

   // Determine largest (max) number
   maxVal = userVals.at(0);         // Largest so far
   for (i = 0; i < userVals.size(); ++i) {
      if (userVals.at(i) > maxVal) {
         maxVal = userVals.at(i);
      }
   }
   cout << "Max: " << maxVal << endl;

   return 0;
}
```

```
Enter 8 integer values...
Value: 3
Value: 5
Value: 23
Value: -1
Value: 456
Value: 1
Value: 6
Value: 83
Max: 456

...

Enter 8 integer values...
Value: -5
Value: -10
Value: -44
Value: -2
Value: -27
Value: -9
Value: -27
Value: -9
Max: -2
```

Feedback?

PARTICIPATION ACTIVITY 5.4.2: Iterating through vectors.

Complete the code provided to achieve the desired goal.

1) Find the minimum element value in vector `valsVctr`.

```
tempVal = valsVctr.at(0);
for (i = 0; i <
valsVctr.size(); ++i) {
   if (valsVctr.at(i) <
                       ) {
      tempVal=
valsVctr.at(i);
   }
}
```

Check Show answer

2) Find the sum of all elements in vector `valsVctr`.

```
valSum =
                       ;
for (i = 0; i <
valsVctr.size(); ++i) {
   valSum += valsVctr.at(i);
}
```

Check Show answer

3) Count the number of negative-valued elements in vector `valsVctr`.

```
numNeg = 0;
for (i = 0; i <
valsVctr.size(); ++i) {
   if (valsVctr.at(i) < 0) {
      numNeg =
                       ;
   }
}
```

Check Show answer

Feedback?

zyDE 5.4.1: Computing the average of a vector's element values.

Complete the code to compute the average of the vector's element values. The re be 16.

Load default template... Run

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

int main() {
   const int VALS_SIZE = 6;
   vector<int> valsVctr(VALS_SIZE);
   unsigned int i;
   int sumVal;
   int avgVal;

   valsVctr.at(0) = 30;
   valsVctr.at(1) = 20;
   valsVctr.at(2) = 20;
   valsVctr.at(3) = 15;
   valsVctr.at(4) = 5;
   valsVctr.at(5) = 10;

   sumVal = 0;
   avgVal = 0;
   /* FIXME: Write for loop to iterate thr
```

Feedback?

Common error: Accessing out of range vector element

A common error is to try to access a vector with an index that is out of the vector's index range. Ex: Trying to access highScores.at(8) when highScores valid indices are 0-7. Care should be taken whenever a user enters a number that is then used as a vector index, and when using a loop index as a vector index also, to ensure the array index is within a vector's valid index range. Accessing an index that is out of range causes the program to automatically abort execution, typically with an error message being automatically printed. For example, for the declaration `vector highScores(8)`, accessing highScores.at(8), or highScores.at(i) where i is 8, yields the following error message when running the program compiled with g++:

Figure 5.4.4: Sample error message when accessing an out of range vector index.

```
terminate called after throwing an instance of 'std::out_of_range'
  what():  vector::_M_range_check
Abort
```

Feedback?

zyDE 5.4.2: Loop expressions.

Run the program, which prints the contents of the vals vector. Modify the program expression to be `i <= VALS_SIZE` rather than `i < VALS_SIZE`, and observe that aborts.

Load default template... Run

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6     const int VALS_SIZE = 6;
7     vector<int> myVals(VALS_SIZE);
8     unsigned int i;
9
10    myVals.at(0) = 30;
11    myVals.at(1) = 20;
12    myVals.at(2) = 20;
13    myVals.at(3) = 15;
14    myVals.at(4) = 5;
15    myVals.at(5) = 10;
16
17    for( i = 0; i < myVals.size(); ++i){
18       cout << "myVals.at(" << i << ") = " <
19    }
20
21
```

Feedback?

PARTICIPATION ACTIVITY 5.4.3: Iterating through a vector.

Given the following code:

```
const int NUM_ELEMENTS = 5;
vector<int> myVctr(NUM_ELEMENTS);
unsigned int i;
```

1) The normal for loop structure

iterates as long as: `i <= myVctr.size()`

- ○ True
- ○ False

2) To compute the sum of elements, a reasonable statement preceding the for loop is: `sumVal = 0;`

- ○ True
- ○ False

3) To find the maximum element value, a reasonable statement preceding the for loop is: `maxVal = 0;`

- ○ True
- ○ False

Feedback?

CHALLENGE ACTIVITY 5.4.1: Enter the output for the vector.

Jump to level 1

Type the program's output.

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

int main() {
   const int NUM_ELEMENTS = 4;
   vector<int> userVals(NUM_ELEMENTS);
   unsigned int i;

   userVals.at(0) = -2;
   userVals.at(1) = 3;
   userVals.at(2) = -6;
   userVals.at(3) = 8;

   for (i = 0; i < userVals.size(); ++i) {
      if (userVals.at(i) < 0) {
         userVals.at(i) = -1 * userVals.at(i);
      }
      cout << userVals.at(i) << endl;
   }

   return 0;
}
```

2
3
6
8

1 2 3 4

Check Next **Done**. Click any level to practice more. Completion is preserv

✔ The for loop iterates in order, once for each element in the vector. If the value of the current element is multiplied by -1. So, the program changes the negative values into positive and outp

Yours

```
2
3
6
8
```

Expected

```
2
3
6
8
```

Feedback?

CHALLENGE ACTIVITY 5.4.2: Finding values in vectors.

Assign numMatches with the number of elements in userValues that equal matchValue. userValues has NUM_VALS elements. Ex: If userValues is {2, 2, 1, 2} and matchValue is 2 , then numMatches should be 3.

Your code will be tested with the following values:
* matchValue: 2, userValues: {2, 2, 1, 2} (as in the example program above)
* matchValue: 0, userValues: {0, 0, 0, 0}
* matchValue: 50, userValues: {10, 20, 30, 40}
(Notes)

```
9   int numMatches = -99; // Assign numMatches with 0 before your for loop
10  vector<int> userValues(NUM_VALS);
11
12  cin >> matchValue;
13
14  for (i = 0; i < userValues.size(); ++i) {
15     cin >> userValues.at(i);
16  }
17
18  /* Your solution goes here  */
19  numMatches =0;
20  for (i = 0; i < userValues.size(); ++i) {
21     if(matchValue == userValues.at(i)){
22        numMatches++;
23     }
```

```
24    }
25
26    cout << "matchValue: " << matchValue << ", numMatches: " << numMatches << endl;
27
28    return 0;
29 }
```

Run ✓ All tests passed

✓ Testing with inputs: 2 2 2 1 2

Your output `matchValue: 2, numMatches: 3`

✓ Testing with inputs: 0 0 0 0 0

Your output `matchValue: 0, numMatches: 4`

✓ Testing with inputs: 50 10 20 30 40

Your output `matchValue: 50, numMatches: 0`

Feedback?

CHALLENGE ACTIVITY 5.4.3: Populating a vector with a for loop.

Write a for loop to populate vector userGuesses with NUM_GUESSES integers. Read integers using cin. Ex: If NUM_GUESSES is 3 and user enters 9 5 2, then userGuesses is {9, 5, 2}.

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

```
19    return 0;
20 }
```

Run ✓ All tests passed

✓ Testing with inputs: 2 4 6

Your output `2 4 6`

✓ Testing with inputs: 8 12 1

Your output `8 12 1`

Feedback?

CHALLENGE ACTIVITY 5.4.4: Vector iteration: Sum of excess.

Vector testGrades contains NUM_VALS test scores. Write a for loop that sets sumExtra to the total extra credit received. Full credit is 100, so anything over 100 is extra credit. Ex: If testGrades = {101, 83, 107, 90}, then sumExtra = 8, because 1 + 0 + 7 + 0 is 8.

```
4
5  int main() {
6     const int NUM_VALS = 4;
7     vector<int> testGrades(NUM_VALS);
8     unsigned int i;
9     int sumExtra = -9999; // Assign sumExtra with 0 before your for loop
10
11    for (i = 0; i < testGrades.size(); ++i) {
12       cin >> testGrades.at(i);
13    }
14
15    /* Your solution goes here  */
16    sumExtra =0;
17    for (i = 0; i < testGrades.size(); ++i) {
18       if(testGrades.at(i) > 100){
19          sumExtra = sumExtra +testGrades.at(i)-100;
20       }
21    }
22
23    cout << "sumExtra: " << sumExtra << endl;
24    return 0;
25 }
```

Run ✓ All tests passed

✓ Testing with inputs: 101 83 107 90

Your output `sumExtra: 8`

✓ Testing with inputs: 100 90 70 50

Your output `sumExtra: 0`

✓ Testing with inputs: 101 110 103 104

Your output `sumExtra: 18`

Feedback?

CHALLENGE ACTIVITY 5.4.5: Printing vector elements separated by commas.

Write a for loop to print all NUM_VALS elements of vector hourlyTemp. Separate elements with a comma and space. Ex: If hourlyTemp = {90, 92, 94, 95}, print:

```
90, 92, 94, 95
```

Your code's output should end with the last element, without a subsequent comma, space, or newline.

```
   vector<int> hourlyTemp(NUM_VALS);

   for (i = 0; i < hourlyTemp.size(); ++i) {
      cin >> hourlyTemp.at(i);
   }

   /* Your solution goes here  */
   for (i = 0; i < hourlyTemp.size(); ++i) {
      if(i !=hourlyTemp.size()-1){
         cout << hourlyTemp.at(i) << ", ";
      }
      else{
         cout << hourlyTemp.at(i);
      }

   }

   cout << endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with inputs: 90 92 94 95

Your output `90, 92, 94, 95`

✓ Testing with inputs: 100 105 90 85

Your output `100, 105, 90, 85`

✓ Testing with inputs: 53 62 45 37

Your output `53, 62, 45, 37`

✓ Testing with inputs: 95 95 95 95

Your output `95, 95, 95, 95`

Feedback?