

# 5.1 Array/vector concept (general)

A typical variable stores one data item, like the number 59 or the character 'a'. Instead, sometimes a *list* of data items should be stored. Ex: A program recording points scored in each quarter of a basketball game needs a list of 4 numbers. Requiring a programmer to declare 4 variables is annoying; 200 variables would be ridiculous. An **array** is a special variable having one name, but storing a list of data items, with each item being directly accessible. Some languages use a construct similar to an array called a **vector**. Each item in an array is known as an **element**.

## PARTICIPATION ACTIVITY

5.1.1: Sometimes a variable should store a list, or array, of data items.



### Animation captions:

1. A variable usually stores just one data item.
2. Some variables should store a list of data items, like variable pointsPerQuarter that stores 4 items.
3. Each element is accessible, like the element numbered 3.

You might think of a normal variable as a truck, and an array variable as a train. A truck has just one car for carrying "data", but a train has many cars, each of which can carry data.

Figure 5.1.1: A normal variable is like a truck, whereas an array variable is like a train.



Source: Truck ([Ian Britton / freefoto.com](#)), Train ([Ian Britton / freefoto.com](#))

BAYLORCSI14301440Spring2021

In an array, each element's location number is called the **index**, myArray[2] has index 2. An array's key feature is that the index enables direct access to any element, as in myArray[2]; different languages

may use different syntax, like myArray(3) or myVector.at(3). In many languages, indices start with 0 rather than 1, so an array with 4 elements has indices 0, 1, 2, and 3.

**PARTICIPATION ACTIVITY**

5.1.2: Update the array's data values.

**Start**

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

Update myItems with the given code.

| myItems |    |
|---------|----|
| 0       | 82 |
| 1       | 4  |
| 2       | 40 |
| 3       | 0  |
| 4       | 63 |
| 5       | 42 |
| 6       | 16 |

**Check****Next****PARTICIPATION ACTIVITY**

5.1.3: Array basics.



Array peoplePerDay has 365 elements, one for each day of the year. Valid accesses are peoplePerDay[0], [1], ..., [364].

- 1) Which assigns element 0 with the value 250?

- peoplePerDay[250] = 0
- peoplePerDay[0] = 250
- peoplePerDay = 250

- 2) Which assigns element 1 with the value 99?

- peoplePerDay[1] = 99
- peoplePerDay[99] = 1

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



- 3) What is the value of peoplePerDay[8]?

```
peoplePerDay[9] = 5;
peoplePerDay[8] = peoplePerDay[9] -
3;
```

- 8
- 5
- 2

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



- 4) Assume N is initially 1. What is the value of peoplePerDay[2]?

```
peoplePerDay[N] = 15;
N = N + 1;
peoplePerDay[N] = peoplePerDay[N -
1] * 3;
```

- 15
- 2
- 45

**PARTICIPATION ACTIVITY**

5.1.4: Arrays with element numbering starting with 0.



Array scoresList has 10 elements with indices 0 to 9, accessed as scoresList[0] to scoresList[9].

- 1) Assign the first element in scoresList with 77.

**Check****Show answer**

- 2) Assign the second element in scoresList with 77.

**Check****Show answer**

- 3) Assign the last element with 77.

**Check****Show answer**

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



- 4) If that array instead has 100 elements, what is the last element's index?



- 5) If the array's last index was 499, how many elements does the array have?



©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

## 5.2 Vectors

### Vector declaration and accessing elements

A programmer commonly needs to maintain a list of items, just as people often maintain lists of items like a grocery list or a course roster. A **vector** is an ordered list of items of a given data type. Each item in a vector is called an **element**. A programmer must include the statement `#include <vector>` at the top of the file when planning to use vectors.

Construct 5.2.1: Vector declaration.

```
vector<dataType> vectorName(numElements);
```

The statement above declares a vector with the specified number of elements, each element of the specified data type. The type of each vector element is specified within the angle brackets (<>). The number of vector elements is specified within parentheses following the vector name. Ex: `vector<int> gameScores(4);` declares a vector `gameScores` with 4 integer elements.

Terminology note: {} are **braces**. <> are **angle brackets**, or **chevrons**. In a vector access, the number in .at() parentheses is called the **index** of the corresponding element. The first vector element is at index 0.

If you have studied arrays, then know that a vector was added to C++ as a safer and more powerful form of arrays, discussed elsewhere.

PARTICIPATION  
ACTIVITY

5.2.1: A vector declaration creates multiple variables in memory, each accessible using .at().



## Animation captions:

1. A vector named itemCounts is declared. The vector consists of 3 elements, each of data type int.
2. An element is accessed with the at() function. The number in parentheses is the index of the corresponding element.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

**PARTICIPATION ACTIVITY**

5.2.2: Vector basics.

Given:

```
vector<int> yearsList(4);  
  
yearsList.at(0) = 1999;  
yearsList.at(1) = 2012;  
yearsList.at(2) = 2025;
```

- 1) How many elements does the vector declaration create?

- 0
- 1
- 3
- 4

- 2) With what value is yearsList.at(1) assigned?

- 1
- 1999
- 2012

- 3) With what value does currYear = yearsList.at(2) assign currYear?

- 2
- 2025
- Invalid index

- 4) Is currYear = yearsList.at(4) a valid assignment?

- Yes, the fourth element is accessed.
- No, yearsList.at(4) does not exist.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021





5) What is the proper way to access the *first* element in vector yearsList?

- yearsList.at(1)
- yearsList.at(0)

6) What are the contents of the vector if the above code is followed by the statement: yearsList.at(0) = yearsList.at(2)?



©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

- 1999, 2012, 1999, 0
- 2012, 2012, 2025, 0
- 2025, 2012, 2025, 0

7) What is the index of the *last* element for the following vector:



```
vector<int> pricesList(100);
```

- 99
- 100
- 101

## Using an expression for a vector index

A powerful aspect of vectors is that the index is an expression. Ex: userNums.at(i) uses the value held in the int variable i as the index. As such, a vector is useful to easily lookup the Nth item in a list.

A vector's index must be an unsigned integer type. The vector index cannot be a floating-point type, even if the value is 0.0, 1.0, etc.

The program below allows a user to print the age of the Nth oldest known person to have ever lived. The program quickly accesses the Nth oldest person's age using `oldestPeople.at(nthPerson - 1)`. Note that the index is `nthPerson - 1` rather than just `nthPerson` because a vector's indices start at 0, so the 1st age is at index 0, the 2nd at index 1, etc.

Figure 5.2.1: Vector's ith element can be directly accessed using .at(i):  
Oldest people program.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

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

int main() {
    vector<int> oldestPeople(5);
    int nthPerson; // User input, Nth
oldest person

    oldestPeople.at(0) = 122; // Died 1997 in France
    oldestPeople.at(1) = 119; // Died 1999 in U.S.
    oldestPeople.at(2) = 117; // Died 1993 in U.S.
    oldestPeople.at(3) = 117; // Died 1998 in Canada
    oldestPeople.at(4) = 116; // Died 2006 in Ecuador

    cout << "Enter N (1..5): ";
    cin >> nthPerson;

    if ((nthPerson >= 1) && (nthPerson <= 5)) {
        cout << "The " << nthPerson << "th oldest person
lived ";
        cout << oldestPeople.at(nthPerson - 1) << "
years." << endl;
    }

    return 0;
}
```

Enter N (1..5): 1  
The 1th oldest person lived 122 years.

...

Enter N (1..5): 4  
The 4th oldest person lived 117 years.

... ©zyBooks 03/11/21 19:10 488201  
xiang zhao  
Enter N (1..5): 9  
BAYLORCSI14301440Spring2021

...

Enter N (1..5): 0

...

Enter N (1..5): 5  
The 5th oldest person lived 116 years.

#### PARTICIPATION ACTIVITY

5.2.3: Nth oldest person program.



- 1) In the program above, what is the purpose of this check:

```
if ((nthPerson >= 1) && (nthPerson
<= 5)) {
    ...
}
```

- To avoid overflow because nthPerson's data type can only store values from 1 to 5.
- To ensure only valid vector elements are accessed because the vector oldestPeople only has 5 elements.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

#### PARTICIPATION ACTIVITY

5.2.4: Vector declaration and accesses.





- 1) Declare a vector named myVals that stores 10 items of type int.

**Check****Show answer**

- 2) Assign x with the value stored at index 8 of vector myVals.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

**Check****Show answer**

- 3) Given myVals has 10 elements, assign the last element in myVals with the value 555.

**Check****Show answer**

- 4) Assign myVals' element at the index held in currIndex with the value 777.

**Check****Show answer**

- 5) Assign tempVal with the myVals' element at the index one after the value held in variable i.

**Check****Show answer**

## Loops and vectors

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

A key advantage of vectors becomes evident when used in conjunction with loops. The program below uses a loop to allow a user to enter 8 integer values, storing those values in a vector, and then printing those 8 values.

A vector's **size()** function returns the number of vector elements. Ex: In the program below, userVals.size() is 8 because the vector was declared with 8 elements.

Figure 5.2.2: Vectors combined with loops are powerful together: User-entered numbers.

```
#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

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

    cout << "You entered: ";
    for (i = 0; i < userVals.size(); ++i) {
        cout << userVals.at(i) << " ";
    }
    cout << endl;

    return 0;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

```
Enter 8 integer values...
Value: 5
Value: 99
Value: -1
Value: -44
Value: 8
Value: 555555
Value: 0
Value: 2
You entered: 5 99 -1 -44 8
555555 0 2
```

#### PARTICIPATION ACTIVITY

#### 5.2.5: Vector with loops.



Refer to the program above.

- 1) How many times does each for loop iterate?

- 1
- 8
- Unknown

- 2) Which one line of code can be changed to allow the user to enter 100 elements?

- `const int NUM_VALS = 8;`
- `for (i = 0; i < userVals.size(); ++i) {`

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

## Vector initialization

A vector's elements are automatically initialized to 0s during the vector declaration.

All of a vector's elements may be initialized to another single value. Ex:

`vector<int> myVector(3, -1);` creates a vector named myVector with three elements, each with value -1.

A programmer may initialize each vector element with different values by specifying the initial values in braces {} separated by commas. Ex: `vector<int> carSales = {5, 7, 11};` creates a vector of three integer elements initialized with values 5, 7, and 11. Such vector declaration and initialization does not require specifying the vector size, because the vector's size is automatically set to the number of elements within the braces. For a larger vector, initialization may be done by first declaring the vector, and then using a loop to assign vector elements.

### PARTICIPATION ACTIVITY

#### 5.2.6: Vector initialization.



- 1) Write a single statement to declare a vector of ints named salesGoals with 4 elements each initialized to 10.

**Check****Show answer**

- 2) Given the following, what is maxScores.at(3)?

```
vector<int> maxScores = {20, 20,  
100, 50};
```

**Check****Show answer**

### Common error: Forgetting to include <vector>

©zyBooks 03/11/21 19:10 488201  
Xiang Zhao  
Baylor CSI 1430/1440 Spring 2021

A common error is to forget the `#include <vector>` at the top of the file when using vectors. Trying to then declare a vector variable may yield a strange compiler error message, such as:

```
testfile.cpp:12: error: ISO C++ forbids declaration of vector with no type  
testfile.cpp:12: error: expected ; before < token
```

The same error message may be seen if the vector library is included but the namespace std is not used.

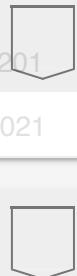
**CHALLENGE ACTIVITY**

5.2.1: Enter the output for the vector.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**CHALLENGE ACTIVITY**

5.2.2: Printing vector elements.



Write three statements to print the first three elements of vector runTimes. Follow each with a newline. Ex: If runTimes = {800, 775, 790, 805, 808}, print:

800  
775  
790

Note: These activities may test code with different test values. This activity will perform two tests, both with a 5-element vector. See "How to Use zyBooks".

Also note: If the submitted code tries to access an invalid vector element, such as runTimes.at(9) for a 5-element vector, 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.

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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**Run**

View your last submission ▾

**CHALLENGE ACTIVITY**

## 5.2.3: Printing vector elements with a for loop.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



Write a for loop to print all NUM\_VALS elements of vector courseGrades, following each with a space (including the last). Print forwards, then backwards. End with newline. Ex: If courseGrades = {7, 9, 11, 10}, print:

```
7 9 11 10
10 11 9 7
```

Hint: Use two for loops. Second loop starts with i = courseGrades.size() - 1 ([Notes](#))

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

Also note: If the submitted code tries to access an invalid vector element, such as courseGrades.at(9) for a 4-element vector, 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.

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

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

**Run**

## 5.3 Arrays

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

### Array declarations and accessing elements

A programmer commonly needs to maintain a list of items, just as people often maintain lists of items like a grocery list or a course roster. An **array** is an ordered list of items of a given data type. Each item in an array is called an **element**.

Construct 5.3.1: Array declaration.

```
dataType arrayName[numElements];
```

This statement declares an array having the specified number of elements in memory, each element of the specified data type. The desired number of elements are specified in [ ] symbols.

Terminology note: [] are **brackets**. {} are **braces**. In an array access, the number in brackets is called the **index** of the corresponding element. The first array element is at index 0.

PARTICIPATION  
ACTIVITY

5.3.1: An array declaration creates multiple variables in memory, each accessible using [].



### Animation captions:

1. An array named itemCounts is declared. The array consists of 3 elements, each of data type int.
2. An element is accessed with brackets. The number in brackets is called the index of the corresponding element.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

PARTICIPATION  
ACTIVITY

5.3.2: Select the index shown.



PARTICIPATION  
ACTIVITY

5.3.3: Array basics.



Given:

```
int yearsArr[4];  
  
yearsArr[0] = 1999;  
yearsArr[1] = 2012;  
yearsArr[2] = 2025;
```

- 1) How many elements in memory does the array declaration create?

- 0
- 1
- 3
- 4

- 2) What value is stored in yearsArr[1]?

- 1
- 1999
- 2012

- 3) With what value does `currYear = yearsArr[2]` assign currYear?

- 2
- 2025
- Invalid index

- 4) With what value does `currYear = yearsArr[3]` assign currYear?

- 3
- 2025
- Invalid index
- Unknown

- 5) Recall that the array declaration was `int yearsArr[4]`. Is `currYear = yearsArr[4]` a valid assignment?

- Yes, it accesses the fourth element.
- No, yearsArr[4] does not exist.

- 6) What is the proper way to access the *first* element in array yearsArr?

- 

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

yearsArr[1]

yearsArr[0]

- 7) What are the contents of the array if the above code is followed by the statement: yearsArr[0] = yearsArr[2]?

1999, 2012, 1999, ?

2012, 2012, 2025, ?

2025, 2012, 2025, ?

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

- 8) What is the index of the *last* element for the following array: `int pricesArr[100];`

99

100

101

## Using an expression for an array index

A powerful aspect of arrays is that the index is an expression. Ex: `userNums[i]` uses the value held in the `int` variable `i` as the index. As such, an array is useful to easily lookup the Nth item in a list.

An array's index must be an integer type. The array index cannot be a floating-point type, even if the value is 0.0, 1.0, etc.

The program below allows a user to print the age of the Nth oldest known person to have ever lived. The program quickly accesses the Nth oldest person's age using `oldestPeople[nthPerson - 1]`. Note that the index is `nthPerson - 1` rather than just `nthPerson` because an array's indices start at 0, so the 1st age is at index 0, the 2nd at index 1, etc.

Figure 5.3.1: Array's nth element can be directly accessed using [n-1]:  
Oldest people program.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

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

int main() {
    int oldestPeople[5];
    int nthPerson; // User input, Nth oldest person

    oldestPeople[0] = 122; // Died 1997 in France
    oldestPeople[1] = 119; // Died 1999 in U.S.
    oldestPeople[2] = 117; // Died 1993 in U.S.
    oldestPeople[3] = 117; // Died 1998 in Canada
    oldestPeople[4] = 116; // Died 2006 in Ecuador

    cout << "Enter N (1..5): ";
    cin >> nthPerson;

    if ((nthPerson >= 1) && (nthPerson <= 5)) {
        cout << "The " << nthPerson << "th oldest person lived ";
        cout << oldestPeople[nthPerson - 1] << " years."
        << endl;
    }

    return 0;
}
```

Enter N (1..5): 1  
The 1th oldest person lived 122 years.

...

Enter N (1..5): 4  
The 4th oldest person lived 117 years.

... ©zyBooks 03/11/21 19:10 488201  
xiang zhao  
Enter N (1..5): 9  
BAYLORCSI14301440Spring2021

...

Enter N (1..5): 0

...

Enter N (1..5): 5  
The 5th oldest person lived 116 years.

#### PARTICIPATION ACTIVITY

#### 5.3.4: Nth oldest person program.



- 1) In the program above, what is the purpose of this if statement:

```
if ((nthPerson >= 1) && (nthPerson
<= 5)) {
    ...
}
```

- To avoid overflow because nthPerson's data type can only store values from 1 to 5.
- To ensure only valid array elements are accessed because the array oldestPeople only has 5 elements.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

#### PARTICIPATION ACTIVITY

#### 5.3.5: Array declaration and accesses.



- 1) Declare an array named myVals that stores 10 items of type int.

**Check****Show answer**

- 2) Assign variable x with the value stored at index 8 of array myVals.

**Check****Show answer**

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

- 3) Assign the second element of array myVals with the value 555.

**Check****Show answer**

- 4) Assign myVals array element at the index held in currIndex with the value 777.

**Check****Show answer**

- 5) Assign tempVal with the myVals array element at the index one after the value held in variable i.

**Check****Show answer**

## Loops and arrays

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

A key advantage of arrays becomes evident when used in conjunction with loops. The program below uses a loop to allow a user to enter 8 integer values, storing those values in an array, and then printing those 8 values.

Figure 5.3.2: Arrays combined with loops are powerful together: User-entered numbers.

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

int main() {
    const int NUM_ELEMENTS = 8; // Number of elements in array
    int userVals[NUM_ELEMENTS]; // User numbers
    int i; // Loop index

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

    cout << "You entered: ";
    for (i = 0; i < NUM_ELEMENTS; ++i) {
        cout << userVals[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Enter 8 integer values...  
 Value: 5  
 Value: 99  
 Value: -1  
 Value: 8  
 Value: 555555  
 Value: 0  
 Value: 2  
 You entered: 5 99 -1 -44 8  
 555555 0 2

#### PARTICIPATION ACTIVITY

#### 5.3.6: Array with loops.



Refer to the program above.

- 1) How many times does each for loop iterate?

- 1
- 8
- Unknown



- 2) Which one line of code can be changed to allow the user to enter 100 elements?

- `const int NUM_ELEMENTS = 8;`
- `for (i = 0; i < NUM_ELEMENTS; ++i) {`



©zyBooks 03/11/21 19:10 488201  
 xiang zhao  
 BAYLORCSI14301440Spring2021

## Array initialization

An array's elements are not automatically initialized during the variable declaration and should be initialized before being read. A programmer may initialize an array's elements in an array variable declaration by specifying the initial values in braces {} separated by commas. Ex:

`int myArray[3] = {5, 7, 11};` initializes element at index 0 with 5, element at index 1 with 7, and element at index 2 with 11. For larger arrays, a loop may be used for initialization.

Like other variables, the keyword **const** may be prepended to an array variable declaration to prevent changes to the array. Thus, `const int YEARS[3] = {1865, 1920, 1964};` followed by `YEARS[0] = 2000;` yields a compiler error.

xiang zhao  
BAYLORCSI14301440Spring2021

#### PARTICIPATION ACTIVITY

5.3.7: Array initialization.



- 1) Declare an array of ints named `myVals` with 4 elements each initialized to 10. The array declaration and initialization should be done in a single statement.

**Check**

**Show answer**

- 2) Given: `int maxScores[4] = {20, 20, 100, 50};` What is `maxScores[3]`?

**Check**

**Show answer**

#### CHALLENGE ACTIVITY

5.3.1: Enter the output for the array.



#### CHALLENGE ACTIVITY

5.3.2: Printing array elements.



Write three statements to print the first three elements of array `runTimes`. Follow each statement with a newline. Ex: If `runTimes = {800, 775, 790, 805, 808}`, print:

xiang zhao  
BAYLORCSI14301440Spring2021

800

775

790

Note: These activities will test the code with different test values. This activity will perform

two tests, both with a 5-element array (int runTimes[5]). See "[How to Use zyBooks](#)".

Also note: If the submitted code tries to access an invalid array element, such as runTime[9] for a 5-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.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int NUM_ELEMENTS = 5;
6     int runTimes[NUM_ELEMENTS];
7     int i;
8
9     for (i = 0; i < NUM_ELEMENTS; ++i) {
10         cin >> runTimes[i];
11     }
12
13     /* Your solution goes here */
14
15     return 0;
16 }
```

**Run**

View your last submission ▾

**CHALLENGE ACTIVITY**

5.3.3: Printing array elements with a for loop.



Write a for loop to print all elements in courseGrades, following each element with a space (including the last). Print forwards, then backwards. End each loop with a newline. Ex: If courseGrades = {7, 9, 11, 10}, print:

```
7 9 11 10
10 11 9 7
```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

Hint: Use two for loops. Second loop starts with i = NUM\_VALS - 1. ([Notes](#))

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

Also note: If the submitted code tries to access an invalid array element, such as `courseGrades[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.

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int NUM_VALS = 4;
6     int courseGrades[NUM_VALS];
7     int i;
8
9     for (i = 0; i < NUM_VALS; ++i) {
10         cin >> courseGrades[i];
11     }
12
13     /* Your solution goes here */
14
15     return 0;
16 }
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

Run

View your last submission ▾

## 5.4 Array/vector iteration drill

The following activities can help one become comfortable with iterating through arrays or vectors, before learning to code such iteration.

### PARTICIPATION ACTIVITY

5.4.1: Find the maximum value in the array.



Click "Store value" if a new maximum value is seen.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

### PARTICIPATION ACTIVITY

5.4.2: Negative value counting in array.



Click "Increment" if a negative value is seen.

**PARTICIPATION ACTIVITY**

5.4.3: Array sorting largest value.



Move the largest value to the right-most position. Click "Swap values" if the larger of the two current values is on the left.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

## 5.5 Iterating through arrays

### Iterating through an array using loops

Iterating through arrays using loops is commonplace and is an important programming skill to master.

Because array indices are numbered 0 to N - 1 rather than 1 to N, programmers commonly use this for loop structure:

Figure 5.5.1: Common for loop structure for iterating through an array.

```
// Iterating through myArray
for (i = 0; i < numElements; ++i) {
    // Loop body accessing myArray[i]
}
```

Note that index variable i is initialized to 0, and the loop expression is  $i < N$  rather than  $i \leq N$ . If N 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 array element is accessed as  $\text{myArray}[i]$  rather than the more complex  $\text{myArray}[i - 1]$ .

**PARTICIPATION ACTIVITY**

5.5.1: Iterating through an array.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



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

```
int daysList[365];  
...  
  
for (i = 0;  
     ; ++i) {  
    cout << daysList[i] <<  
    endl;  
}
```

**Check****Show answer**

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

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

```
for (i = 0; i < 99; ++i) {  
    cout << someArray[i] << endl;  
}
```

**Check****Show answer**

## Determining a quantity about an array's items

Iterating through an array for various purposes is an important programming skill to master. The program below computes the sum of an array's element values. For computing the sum, the program initializes a variable sum to 0, then simply adds the current iteration's array element value to that sum.

Figure 5.5.2: Iterating through an array example: Program that computes the sum of an array's elements.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

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

int main() {
    const int NUM_ELEMENTS = 8; // Number of elements
    int userVals[NUM_ELEMENTS]; // User numbers
    int i; // Loop index
    int sumVal; // For computing sum

    // Prompt user to populate array
    cout << "Enter " << NUM_ELEMENTS << " integer values..." << endl;

    for (i = 0; i < NUM_ELEMENTS; ++i) {
        cout << "Value: ";
        cin >> userVals[i];
    }

    // Determine sum
    sumVal = 0;
    for (i = 0; i < NUM_ELEMENTS; ++i) {
        sumVal = sumVal + userVals[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
xian zhao
BAYLORCSI14301440Spring2021
Enter 8 integer
values...
Value: 3
Value: 5
Value: 234
Value: 346
Value: 234
Value: 73
Value: 26
Value: 1
Sum: 922
```

## Finding the maximum value in an array

Programs commonly iterate through arrays to determine some quantity about the array's items. 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 "so far" as the program iterates through the array. During each iteration, if the array's current element value is larger than the max seen so far, the program assigns maxVal with the array element.

Before entering the loop, maxVal must be initialized to some value because max will be compared with each array 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 an array element (using the first element, though any element could have been used) and initializes maxVal to that element's value.

Figure 5.5.3: Iterating through an array example: Program that finds the max item.

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

int main() {
    const int NUM_ELEMENTS = 8; // Number of elements
    int userVals[NUM_ELEMENTS]; // Array of user numbers
    int i; // Loop index
    int maxVal; // Computed max

    // Prompt user to populate array
    cout << "Enter " << NUM_ELEMENTS << " integer values..." << endl;

    for (i = 0; i < NUM_ELEMENTS; ++i) {
        cout << "Value: ";
        cin >> userVals[i];
    }

    // Determine largest (max) number
    maxVal = userVals[0]; // Largest so far

    for (i = 0; i < NUM_ELEMENTS; ++i) {
        if (userVals[i] > maxVal) {
            maxVal = userVals[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

©zyBooks 03/11/21 19:10 488201  
 xiang zhao  
 BAYLORCSI14301440Spring2021

Enter 8 integer values...

Value: -5  
 Value: -10  
 Value: -44  
 Value: -2  
 Value: -27  
 Value: -9  
 Value: -27  
 Value: -9  
 Max: -2

**PARTICIPATION ACTIVITY**
**5.5.2: Array iteration.**


Given an integer array myVals of size N\_SIZE (i.e. int myVals[N\_SIZE] ), complete the code to achieve the stated goal.

- 1) Determine the minimum number in the array, using the same initialization as the maximum number example above.



```
minVal = [ ] ;
for (i = 0; i < N_SIZE; ++i) {
    if (myVals[i] < minVal) {
        minVal = myVals[i];
    }
}
```

©zyBooks 03/11/21 19:10 488201  
 xiang zhao  
 BAYLORCSI14301440Spring2021

**Check**
**Show answer**


- 2)

Count how many negative numbers exist in the array.

```
cntNeg = 0;
for (i = 0; i < N_SIZE; ++i) {
    if ( [ ] ) {
        ++cntNeg;
    }
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**Check**

[Show answer](#)

- 3) Count how many odd numbers exist in the array.

```
cntOdd = 0;
for (i = 0; i < N_SIZE; ++i) {
    if ( (myVals[i] % 2) == 1 )
    {
        [ ];
    }
}
```

**Check**

[Show answer](#)

### zyDE 5.5.1: Print the sum and average of an array's elements.

Modify the program to print the average (mean) as well as the sum. Hint: You don't actually have to change the loop, but rather change what you print.

[Load default template...](#)

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int NUM_ELEMENTS = 8; // Number of elements
6     int userVals[NUM_ELEMENTS]; // User input values
7     int i; // Loop counter
8     int sumVal; // For calculating sum
9
10    // Prompt user to populate array
11    cout << "Enter " << NUM_ELEMENTS << endl;
12
13    for (i = 0; i < NUM_ELEMENTS; ++i) {
14        cout << "Value: " << endl;
15        cin >> userVals[i];
16    }
```

3 5 234 346 234 73 26 -1

**Run**

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

## zyDE 5.5.2: Print selected elements of an array.

Modify the program to instead just print each number that is greater than 21.

The screenshot shows the zyDE (ZyBooks Development Environment) interface. On the left is a code editor with the following C++ code:

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int NUM_ELEMENTS = 8; // Number of elements
6     int userVals[NUM_ELEMENTS]; // User values
7     int i; // Loop index
8     int sumVal; // Sum of values
9
10    // Prompt user to populate array
11    cout << "Enter " << NUM_ELEMENTS << endl;
12
13    for (i = 0; i < NUM_ELEMENTS; ++i) {
14        cout << "Value: " << endl;
15        cin >> userVals[i];
16    }
17
18

```

On the right is a terminal window showing the output of the program. The user has entered the numbers 3, 5, 2, 3, 4, 3, 4, 6, 2, 3, 4, 7, 3, 2, 6, -1. The program has printed the numbers 3, 5, 2, 3, 4, 3, 4, 6, 2, 3, 4, 7, 3, 2, 6, which are all greater than 21.

**Run**

## Common error: Accessing out of range array element

A common error is to try to access an array with an index that is out of the array's index range. Ex: Trying to access `highScores[8]` when `highScores`'s valid indices are 0-7. Care should be taken whenever a user enters a number that is then used as an array index, and when using a loop index as an array index also, to ensure the index is within the array's valid index range. Checking whether an array index is in range is very important. Trying to access an array with an out-of-range index is not only a very common error, but is also one of the hardest errors to debug. The following animation shows what happens when a program writes to an out-of-range index using an array.

PARTICIPATION  
ACTIVITY

5.5.3: Writing to an out-of-range index using an array.

©zyBooks 03/11/21 19:10 488201  
xiang zhao

BAYLORCSI14301440Spring2021

### Animation captions:

1. Variable `userAge` is allocated a location in memory immediately after the array `userWeights`. `userAge` is assigned with 44.

2. Each element of array userWeights is assigned a value.
3. 3 is out of userWeights's index range, and results in overwriting userAge's value.

A write to an array with an out-of-range index may simply write to a memory location of a different variable X residing next to the array in memory. Later, when the program tries to read X, the program encounters incorrect data. For example, a program may write X with the number 44, but when reading X later in the program X may be 2533, with X never (intentionally) written by any program statement in between.

xiang zhao

BAYLORCSI14301440Spring2021

**PARTICIPATION ACTIVITY**

5.5.4: Iterating through an array.



Given the following code:

```
const int NUM_ELEMENTS = 5;  
int myArray[NUM_ELEMENTS];  
int i;
```

- 1) The normal for loop structure iterates

as long as:

```
i <= NUM_ELEMENTS
```

- 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

**CHALLENGE ACTIVITY**

5.5.1: Enter the output for the array.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**CHALLENGE ACTIVITY**

5.5.2: Finding values in arrays.



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

Your code will be tested with the following values:

matchValue: 2, userValues: {2, 1, 2, 2} (as in the example program above)

matchValue: 0, userValues: {0, 0, 0, 0}

matchValue: 10, userValues: {20, 50, 70, 100}

(Notes)

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int NUM_VALS = 4;
6     int userValues[NUM_VALS];
7     int i;
8     int matchValue;
9     int numMatches = -99; // Assign numMatches with 0 before your for loop
10
11    cin >> matchValue;
12    for (i = 0; i < NUM_VALS; ++i) {
13        cin >> userValues[i];
14    }
15
16    /* Your solution goes here */
17
18    cout << "matchValue: " << matchValue << ", numMatches: " << numMatches << endl;

```

**Run**

View your last submission ▾

**CHALLENGE  
ACTIVITY**

5.5.3: Populating an array with a for loop.



Write a for loop to populate array 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}.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int NUM_GUESSES = 3;
6     int userGuesses[NUM_GUESSES];
7     int i;
8
9     /* Your solution goes here */

```

```
10 for (i = 0; i < NUM_GUESSES; ++i) {  
11     cout << userGuesses[i] << " "  
12 }  
13  
14 return 0;  
15 }  
16 }
```

**Run**

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

View your last submission ▾

**CHALLENGE ACTIVITY****5.5.4: Array iteration: Sum of excess.**

Array 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.

```
1 #include <iostream>  
2 using namespace std;  
3  
4 int main() {  
5     const int NUM_VALS = 4;  
6     int testGrades[NUM_VALS];  
7     int i;  
8     int sumExtra = -9999; // Assign sumExtra with 0 before your for loop  
9  
10    for (i = 0; i < NUM_VALS; ++i) {  
11        cin >> testGrades[i];  
12    }  
13  
14    /* Your solution goes here */  
15  
16    cout << "sumExtra: " << sumExtra << endl;  
17    return 0;  
18 }
```

**Run**

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

View your last submission ▾

**CHALLENGE ACTIVITY****5.5.5: Printing array elements separated by commas.**

Write a for loop to print all NUM\_VALS elements of array 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.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int NUM_VALS = 4;
6     int hourlyTemp[NUM_VALS];
7     int i;
8
9     for (i = 0; i < NUM_VALS; ++i) {
10         cin >> hourlyTemp[i];
11     }
12
13     /* Your solution goes here */
14
15     cout << endl;
16
17     return 0;
18 }
```

**Run**

View your last submission ▾

## 5.6 Iterating through vectors

### Iterating through vectors using loops

©zyBooks 03/11/21 19:10 488201

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 5.6.1: 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)
}
```

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**
**5.6.1: 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;
    [REDACTED]; ++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**

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

## 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 5.6.2: 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;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

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

## 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 peeked 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 5.6.3: 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 values..." << endl;©zy
    for (i = 0; i < userVals.size(); ++i) {
        cout << "Value: ";
        cin >> userVals.at(i);
    }

    // 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

Books 03/11/21 19:10 488201  
 ... xiang zhao  
 BAYLORCSI14301440Spring2021

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

**PARTICIPATION ACTIVITY**

### 5.6.2: 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);
    }
}
```

©zyBooks 03/11/21 19:10 488201  
 xiang zhao  
 BAYLORCSI14301440Spring2021

**Check**

**Show answer**

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

```
valSum = [REDACTED];  
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`.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

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

**Check****Show answer**

## zyDE 5.6.1: Computing the average of a vector's element values.

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

**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> valsVctr(VALS_SIZE);  
8     unsigned int i;  
9     int sumVal;  
10    int avgVal;  
11  
12    valsVctr.at(0) = 30;  
13    valsVctr.at(1) = 20;  
14    valsVctr.at(2) = 20;  
15    valsVctr.at(3) = 15;  
16    valsVctr.at(4) = 5;  
17    valsVctr.at(5) = 10;
```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

## 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 5.6.4: 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
```

### zyDE 5.6.2: Loop expressions.

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

```

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        ...
19    }
20}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**PARTICIPATION ACTIVITY**

## 5.6.3: Iterating through a vector.



Given the following code:

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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

- 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: `int sumVal = 0;`

- True  
 False

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

- True  
 False

**CHALLENGE ACTIVITY**

## 5.6.1: Enter the output for the vector.

**CHALLENGE ACTIVITY**

## 5.6.2: 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)

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     const int NUM_VALS = 4;
7     int matchValue;
8     unsigned int i;
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 */

```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**Run**

View your last submission ▾

**CHALLENGE ACTIVITY**

5.6.3: 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
12    for (i = 0; i < userGuesses.size(); ++i) {
13        cout << userGuesses.at(i) << " ";
14    }
15
16    return 0;
17 }

```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**Run**

View your last submission ▾

**CHALLENGE ACTIVITY****5.6.4: Vector iteration: Sum of excess.**

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

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.

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
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
17    cout << "sumExtra: " << sumExtra << endl;
18    return 0;
```

**Run**

View your last submission ▾

**CHALLENGE ACTIVITY****5.6.5: 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:

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

90, 92, 94, 95

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

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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

Run

View your last submission ▾

## 5.7 Multiple vectors

Programmers commonly use multiple same-sized vectors to store related lists. The program below maintains a list of country names, and another list indicating average minutes of TV watched per day in each corresponding country.

The statement `if (ctryNames.at(i) == userCountry)` compares the current `ctryNames` element with the user-entered country name. If the names match, the program prints the `ctryMins` element at the same index.

The loop's expression `(i < ctryNames.size()) && (!foundCountry)` depends on the value of the variable `foundCountry`. This expression prevents the loop from iterating through the entire vector once the correct country is found.

The program's numbers aren't made up, by the way: Americans watch nearly 5 hours of TV per day on average.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

Figure 5.7.1: Multiple vector example: TV watching time program.

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

int main() {
    // Source: www.statista.com, 2015
    const int NUM_COUNTRIES = 5; // Num countries supported
    vector<string> ctryNames(NUM_COUNTRIES); // Country names
    vector<int> ctryMins(NUM_COUNTRIES); // Mins TV watched daily
    string userCountry; // User defined country
    bool foundCountry = false; // Match to country supported
    unsigned int i; // Loop index

    // Fill vector contents
    ctryNames.at(0) = "China";
    ctryMins.at(0) = 155;

    ctryNames.at(1) = "Sweden";
    ctryMins.at(1) = 154;

    ctryNames.at(2) = "Russia";
    ctryMins.at(2) = 246;

    ctryNames.at(3) = "UK";
    ctryMins.at(3) = 216;

    ctryNames.at(4) = "USA";
    ctryMins.at(4) = 274;

    // Prompt user for country name
    cout << "Enter country name: ";
    cin >> userCountry;

    // Find country's index and average TV time
    foundCountry = false;
    for (i = 0; (i < ctryNames.size()) && (!foundCountry); ++i) {
        if (ctryNames.at(i) == userCountry) {
            foundCountry = true;
            cout << "People in " << userCountry << " watch ";
            cout << ctryMins.at(i) << " mins of TV daily." << endl;
        }
    }
    if (!foundCountry) {
        cout << "Country not found; try again." << endl;
    }

    return 0;
}
```

Enter country name: USA  
 People in USA watch 274 mins of TV daily.

...

Enter country name: Sweden  
 People in Sweden watch 154 mins of TV daily.

...

©zyBooks 03/11/21 19:10 488201  
 xiang zhao  
 Enter country name: Brazil  
 Country not found; try again.

Consider the above TV watching program involving multiple vectors.

- 1) Multiple vectors saved memory over using one larger vector.

True

False

- 2) Each vector should be the same data type.

True

False

- 3) Each vector should have the same number of elements.

True

False

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



### zyDE 5.7.1: Improve the TV watching time program.

Modify the program such that if a user types a country name that isn't found, print a list known countries.

Load default template...

```
1 #include <iostream>
2 #include <vector>
3 #include <string>
4 using namespace std;
5
6 int main() {
7     // Source: www.statista.com, 2015
8     const int NUM_COUNTRIES = 5;
9     vector<string> ctryNames(NUM_COUNTRI
10    vector<int> ctryMins(NUM_COUNTRIE
11    string userCountry;
12    bool foundCountry = false;
13    unsigned int i;
14
15    // Fill vector contents
16    ctryNames.at(0) = "China";
17    ctryMins.at(0) = 155;
```

USA

Run

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

CHALLENGE

<https://learn.zybooks.com/zybook/BAYLORCSI14301440Spring2021/chapter/5/print>

42/104

**ACTIVITY**

## 5.7.1: Printing the sum of two vector elements.



Add each element in origList with the corresponding value in offsetAmount. Print each sum followed by a space. Ex: If origList = {40, 50, 60, 70} and offsetAmount = {5, 7, 3, 0}, print:

45 57 63 70

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     const int NUM_VALS = 4;
7     vector<int> origList(NUM_VALS);
8     vector<int> offsetAmount(NUM_VALS);
9     unsigned int i;
10
11    for (i = 0; i < origList.size(); ++i) {
12        cin >> origList.at(i);
13    }
14
15    for (i = 0; i < offsetAmount.size(); ++i) {
16        cin >> offsetAmount.at(i);
17    }
18

```

**Run**

View your last submission ▾

**CHALLENGE ACTIVITY**

## 5.7.2: Multiple vectors: Key and value.



For any element in keysList with a value greater than 100, print the corresponding value in itemsList, followed by a space. Ex: If keysList = {42, 105, 101, 100} and itemsList = {10, 20, 30, 40}, print:

20 30

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

Since keysList.at(1) and keysList.at(2) have values greater than 100, the value of itemsList.at(1) and itemsList.at(2) are printed.

```

1 #include <iostream>
2 #include <vector>

```

```
3 using namespace std;
4
5 int main() {
6     const int SIZE_LIST = 4;
7     vector<int> keysList(SIZE_LIST);
8     vector<int> itemsList(SIZE_LIST);
9     unsigned int i;
10
11    for (i = 0; i < keysList.size(); ++i) {
12        cin >> keysList.at(i);
13    }
14
15    for (i = 0; i < itemsList.size(); ++i) {
16        cin >> itemsList.at(i);
17    }
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

Run

View your last submission ▾

## 5.8 Vector resize

Commonly, the size of a list of items is not known during a program's compile time. Thus, a vector's size need not be specified in the vector's declaration. Instead, a vector's size can be set or changed while a program executes using **resize(N)**. Ex: `highScore.resize(10)` resizes the `highScores` vector to have 10 elements.

`resize()` can be called multiple times. If the new size is larger, `resize()` adds elements at the end. If smaller, `resize()` deletes elements from the end. If `userScores` has size 3 (elements 0, 1, 2), `userScores.resize(2);` would delete element 2, leaving elements 0 and 1. A subsequent access to `userScores.at(2)` would result in an error.

PARTICIPATION ACTIVITY

5.8.1: Vector resize.



### Animation captions:

1. When initially declared, the vector `carSales` has a size of 0.
2. Accessing any element, such as `carSales.at(0)`, would result in a runtime error.
3. `carSales.resize(3)` allocates 3 elements with indices 0, 1, and 2. Elements can now be accessed using the `.at()` function.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

The program below asks a user to indicate the number of values the user will enter, allocates that number of elements for a vector, assigns the vector's elements with user-entered values, and then

displays the vector's elements.

Figure 5.8.1: Resizing a vector based on user input.

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

int main() {
    vector<int> userVals; // No elements yet
    int numVals;
    unsigned int i;

    cout << "Enter number of integer values: ";
    cin >> numVals;

    userVals.resize(numVals); // Allocate elements

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

    cout << "You entered: ";
    for (i = 0; i < userVals.size(); ++i) {
        cout << userVals.at(i) << " ";
    }
    cout << endl;

    return 0;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

```
Enter number of integer
values: 7
Enter 7 integer values...
Value: -5
Value: -99
Value: 0
Value: 13
Value: 7
Value: -22
Value: 1
You entered: -5 -99 0 13 7
-22 1
```

#### PARTICIPATION ACTIVITY

#### 5.8.2: Vector resize and size functions.

Given the vector declaration:

```
vector<int> agesVctr;
```

- 1) Immediately after the declaration,  
agesVctr has only 1 element.

- True
- False

- 2) agesVctr.size(4) allocates 4 elements  
for agesVctr.

- True
- False

- 3)

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

Given agesVctr has 3 elements, agesVctr.resize(4) adds 4 more elements, totalling 7 elements.

- True
- False

- 4) Given agesVctr has 3 elements with values 22, 18, and 19, agesVctr.resize(2) changes agesVctr to have 2 elements with values 22 and 18.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

- True
- False

- 5) After agesVctr.resize(5) and agesVctr.at(0) = 99, agesVctr.size() evaluates to 1.

- True
- False

**CHALLENGE ACTIVITY**

5.8.1: Determining the size of a vector.



Assign currentSize with the size of the sensorReadings vector.

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     vector<int> sensorReadings(4);
7     int currentSize;
8     int input;
9     cin >> input;
10    sensorReadings.resize(input);
11
12    /* Your solution goes here */
13
14    cout << "Number of elements: " << currentSize << endl;
15
16    return 0;
17 }
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**Run**

View your last submission ▾

**CHALLENGE  
ACTIVITY****5.8.2: Resizing a vector.**

Resize vector `countDown` to have `newSize` elements. Populate the vector with integers  $\{newSize, newSize - 1, \dots, 1\}$ . Ex: If `newSize = 3`, then `countDown = \{3, 2, 1\}`, and the sample program outputs:

3 2 1 Go!

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

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

**Run**

View your last submission ▾

## 5.9 Vector `push_back`

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

### Appending items to a vector

A programmer can append a new element to the end of an existing vector using a vector's **`push_back()`** function. Ex: `dailySales.push_back(521)` creates a new element at the end of the vector `dailySales` and assigns that element with the value 521.

**PARTICIPATION ACTIVITY**

5.9.1: The vector push\_back() function.

**Animation captions:**

1. When initially declared, the vector dailySales has a size of 0.
2. The push\_back() function appends a new element to the vector.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

**PARTICIPATION ACTIVITY**

5.9.2: Vector push\_back().



- 1) If vector itemPrices has two elements with values 45, 48, what does `itemPrices.size()` return?

**Check****Show answer**

- 2) If itemPrices has element values 45, 48, then after `itemPrices.push_back(38)`, what are itemPrices' element values? Type answer as: 50, 60, 70

**Check****Show answer****Vector pop\_back() and back()**

The following table summarizes a few common functions dealing with the back (or last element) of a vector.

Table 5.9.1: Functions on the back of a vector.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

|                    |   |  |
|--------------------|---|--|
| <b>push_back()</b> | <pre>void push_back(const int newVal);</pre> <p>Append new element having value newVal.</p> | <pre>// playersList initially<br/>55, 99, 44 (size is 3)<br/>playersList.push_back(77);<br/>// Appends new element 77<br/>// playersList is now 55,<br/>99, 44, 77 (size is 4)</pre> |
|                    |   |  |

|                   |   |  |
|-------------------|---|--|
| <b>back()</b>     | <pre>int back();</pre> <p>Returns vector's last element. Vector is unchanged.</p> | <pre>// playersList initially<br/>55, 99, 44<br/>cout &lt;&lt;<br/>playersList.back(); //<br/>Prints 44<br/>// playersList is still<br/>55, 99, 44</pre>   |
| <b>pop_back()</b> | <pre>void pop_back();</pre> <p>Removes the last element.</p>                      | <pre>// playersList is 55, 99,<br/>44 (size 3)<br/>playersList.pop_back(); //<br/>Removes last element<br/>// playersList now 55, 99<br/>(size 2)</pre> <pre>cout &lt;&lt;<br/>playersList.back(); //<br/>Common combination of<br/>back()<br/>playersList.pop_back();<br/>// followed by pop_back()<br/>// Prints 99. playersList<br/>becomes just 55</pre> <pre>cout &lt;&lt;<br/>playersList.pop_back(); //<br/>Common error:</pre> <pre>// pop_back() returns void</pre> |

Shown for vector<int>, but applies to other types.

The program below declares a vector groceryList, which is initially empty. As the user enters grocery items one at a time, the program uses push\_back() to append the items to the list. When done, the user can go shopping, and is presented one list item at a time (which the user presumably finds and places in a shopping cart). The program uses back() to get each item from the list and pop\_back() to remove the item from the list. When the list is empty, shopping is finished.

Note that because the program removes items from the end of the list, the items are presented in reverse order.

Figure 5.9.1: Using push\_back(), back(), and pop\_back(): A grocery list example.

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

int main() {
    vector<string> groceryList; // Vector storing
shopping list
    string groceryItem; // Individual grocery
items
    string userCmd; // User input

    // Prompt user to populate shopping list
    cout << "Enter grocery items or type done." << endl;
    cin >> groceryItem;
    while (groceryItem != "done") {
        groceryList.push_back(groceryItem);
        cin >> groceryItem;
    }

    // Display shopping list
    cout << endl << "Enter any key for next item." <<
endl;
    while (groceryList.size() > 0) {
        groceryItem = groceryList.back();
        groceryList.pop_back();
        cout << groceryItem << " ";
        cin >> userCmd;
    }
    cout << endl << "Done shopping." << endl;

    return 0;
}
```

Enter grocery items or type  
done.  
Oranges  
Apples  
Bread  
Juice  
done

Enter any key for next item.  
Juice a  
Bread a  
Apples a xiang zhao  
Oranges a  
Done shopping.

### PARTICIPATION ACTIVITY

5.9.3: Vector back() and pop\_back() functions.



- 1) If itemPrices has elements 45, 22, 38,  
what does price =  
itemPrices.pop\_back() assign to price?  
Type Error if appropriate.


  
Check    Show answer

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

- 2) If itemPrices has elements 45, 22, 38,  
what does price = itemPrices.back()  
assign to price? Type Error if  
appropriate.



- 3) If itemPrices has elements 45, 22, 38, what is the vector after itemPrices.back() is called? Type answer as: 50, 60, 70



©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

- 4) If itemPrices has elements 45, 22, 38, then after itemPrices.pop\_back(), what does itemPrices.at(2) return? Type Error if appropriate.



**CHALLENGE ACTIVITY**

5.9.1: Appending a new element to a vector.



Append newValue to the end of vector tempReadings. Ex: If newValue = 67, then tempReadings = {53, 57, 60} becomes {53, 57, 60, 67}.

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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

View your last submission ▾

**CHALLENGE ACTIVITY**

5.9.2: Removing an element from the end of a vector.



Remove the last element from vector ticketList.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

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

**Run**

View your last submission ▾

**CHALLENGE ACTIVITY**

5.9.3: Reading the vector's last element.



Write a statement to print "Last mpg reading: " followed by the value of mpgTracker's last element. End with newline. Ex: If mpgTracker = {17, 19, 20}, print:

Last mpg reading: 20

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     const int NUM_ELEMENTS = 3;
```

```
7  vector<int> mpgTracker(NUM_ELEMENTS);
8  int i;
9
10 for (i = 0; i < mpgTracker.size(); ++i) {
11     cin >> mpgTracker.at(i);
12 }
13
14 /* Your solution goes here */
15
16 return 0;
17 }
```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

**Run**

View your last submission ▾

## 5.10 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 5.10.1: Modifying a vector during iteration example: Converting negatives to 0.

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

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

```
#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;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

#### PARTICIPATION ACTIVITY

#### 5.10.1: Modifying a vector in a loop.



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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021



## zyDE 5.10.1: Modifying a vector during iteration example: Doubling element values.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

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() {

```

67 -5 -99 4 22

Run

```
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 <<
12 for (i = 0; i < userVals.size(); ++i
13     cout << "Value: " << endl;
14     cin >> userVals.at(i);
15 }
16
17 /* Type your code here */
18
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

## 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 5.10.2: Using = to copy a vector: Original and sale prices.

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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

```
#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;
}
```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

**PARTICIPATION ACTIVITY**

## 5.10.2: 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**

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



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**

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



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

What size is newVals after:

```
newVals = userVals;
```

...

```
newVals = otherVals;
```

If appropriate type: Error

**Check**

**Show answer**

## 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

5.10.3: Vector comparing.



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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021



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:

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

- True
- False

**CHALLENGE ACTIVITY**

5.10.1: 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}.

```
1 #include <iostream>
2 #include <vector>
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
16    for (i = 0; i < lowerScores.size(); ++i) {
17        cout << lowerScores.at(i) << " ";
18    }
```

Run

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

View your last submission ▾

**CHALLENGE ACTIVITY**

5.10.2: 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.

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
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
17    for (i = 0; i < newScores.size(); ++i) {
18        cout << newScores.at(i) << " ";

```

**Run**

View your last submission ▾

**CHALLENGE ACTIVITY**

5.10.3: 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
16    for (i = 0; i < bonusScores.size(); ++i) {
17        cout << bonusScores.at(i) << " ";
18    }
19}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

Run

View your last submission ▾

**CHALLENGE ACTIVITY**

5.10.4: Modify a vector's elements.



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

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
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}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

Run

View your last submission ▾

**CHALLENGE ACTIVITY****5.10.5: 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}.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     const int NUM_POINTS = 4;
7     vector<int> oldData(NUM_POINTS);
8     vector<int> newData(NUM_POINTS);
9     unsigned int i;
10
11    for (i = 0; i < oldData.size(); ++i) {
12        cin >> oldData.at(i);
13    }
14
15    for (i = 0; i < newData.size(); ++i) {
16        cin >> newData.at(i);
17    }
18}
```

**Run**

View your last submission ▾

## 5.11 Swapping two variables (General)

Sometimes a program must swap values among two variables. **Swapping** two variables x and y means to assign y's value to x, and x's value to y. If x is 33 and y is 55, then after swapping x is 55 and y is 33.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

A common method for swapping uses a temporary variable. A **temporary variable** is a variable used briefly to store a value. To understand the intuition of such temporary storage, consider a person holding a book in one hand and a phone in the other, wishing to swap the items. The person can temporarily place the phone on a table, move the book to the other hand, then pick up the phone.

**PARTICIPATION ACTIVITY****5.11.1: Swap idea: Use a temporary location.**

## Animation captions:

1. A swap between two hands requires a third, temporary place

Similarly, swapping two variables can use a third variable to temporarily hold one value while the other value is copied over.

PARTICIPATION  
ACTIVITY

5.11.2: Swapping two variables using a third temporary variable

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

## Animation captions:

1. 55 is assigned to x. 33 is lost. The swap fails because no temporary variable was used.
2. The swap succeeds because x's value is saved before being assigned 55, and then 33 is assigned to y.

PARTICIPATION  
ACTIVITY

5.11.3: Swap.

To begin, x is 22 and y is 99. What are x and y after the given code?

1) `x = y;`  
`y = x;`

- x is 99 and y is 22.
- x is 22 and y is 99.
- x is 99 and y is 99.

2) `x = y;`  
`y = x;`  
`x = y;`

- x is 99 and y is 22.
- x is 99 and y is 99.
- x is 22 and y is 22.

3) `tempVal = x;`  
`x = y;`  
`y = tempVal;`

- x is 99 and y is 22.
- x is 99 and y is 99.

4) `tempVal = x;`  
`x = y;`  
`y = tempVal;`

- x is 99 and y is 22.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

x is 99 and y is 99.

If you have studied arrays or vectors (or other kinds of lists), know that most swaps are actually performed between two list elements. For example, reversing a list with N elements can be achieved by swapping element 1 and N, element 2 and N-1, element 3 and N-2, etc. (stopping at the middle of the list).

**PARTICIPATION ACTIVITY**

5.11.4: Reversing a list using swaps.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021**Animation captions:**

1. Swap outermost elements.
2. Swap next outermost elements, repeat until reach at middle.

**PARTICIPATION ACTIVITY**

5.11.5: Reversing a list using swaps.

- 1) Using the above approach, how many swaps are needed to reverse this list:  
999 888 777 666 555 444 333 222



## 5.12 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. 19:10 488201  
xiang zhao

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 5.12.1: 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;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

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

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 5.12.2: 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;
}
```

©zyBooks 03/11/21 19:10 488201

Xiang Zhao  
 Enter 8 integer values...  
 Spring 2021  
 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

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(i) so we don't lose that element's value.

Figure 5.12.3: 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  
xiang zhao

BAYLORCSI14301440Spring2021

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 5.12.4: 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;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

```
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
```

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

5.12.1: 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;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

```
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);
}
```

## 5.13 Arrays vs. vectors

C++ supports two kinds of ordered list types.

- **Arrays:** declared as `int myList[10]`, accessed as `myList[i]`.
- **Vectors:** declared as `vector<int> myList(10)`, accessed as `myList.at(i)`.

Arrays have a simpler syntax than vectors, but vectors are safer to use. Thus, using vectors rather than arrays is good practice.

Vectors are safer because the access `v.at(i)` is checked during execution to ensure the index is within the vector's valid range. An array access `a[i]` involves no such check. Such checking is important; trying to access an array with an out-of-range index is a very common error, and one of the hardest errors to debug.

### PARTICIPATION ACTIVITY

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

5.13.1: Writing to an out-of-range index using an array.

### Animation captions:

1. Variable `userAge` is allocated a location in memory immediately after the array `userWeights`. `userAge` is assigned with 44.
2. Each element of array `userWeights` is assigned a value.

3. 3 is out of userWeights's index range, and results in overwriting userAge's value.

As shown above, assigning with an out-of-range index can mysteriously change some other variable's value. Debugging such an error can be a nightmare.

Vectors have more advantages, like resizing during runtime, easy insertion of items at the front or rear, determining vector size, etc., discussed later. Arrays have minor benefits that don't really outweigh drawbacks. Like choosing to not wear seatbelts, choosing to not use vectors may be quite risky.

C++ allows vectors to be accessed using brackets [], but brackets involve no range checking, so a good practice is to use .at() to access vector elements.



**PARTICIPATION ACTIVITY**

5.13.2: Arrays and vectors.



Given:

```
int arrayList[5];
vector<int> vectorList(5);
```

1) arrayList[6] = 777 will yield a compiler error.



- True
- False

2) vectorList[6] = 777 will yield a compiler error.



- True
- False

3) arrayList[6] = 777 will execute without an error message.



- True
- False

4) vectorList.at(6) = 777 will execute without an error message.



- True
- False

5) vectorList[6] = 777 will execute without an error message.



True False6) `while ( i < arrayList.size() )`

) loops while i is less than the array's size.

 True False

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

## 5.14 Two-dimensional arrays

An array can be declared with two dimensions. `int myArray[R][C]` represents a table of int variables with R rows and C columns, so  $R \times C$  elements total. For example, `int myArray[2][3]` creates a table with 2 rows and 3 columns, for 6 int variables total. Example accesses are `myArray[0][0] = 33;` or `num = myArray[1][2].`

### PARTICIPATION ACTIVITY

5.14.1: Two-dimensional array.

#### Animation captions:

1. Conceptually, a two-dimensional array is a table with rows and columns.
2. A two-dimensional array is implemented in a one-dimensional memory by placing each row following the previous row.

Conceptually, a two-dimensional array is a table with rows and columns. The compiler maps two-dimensional array elements to one-dimensional memory, each row following the previous row, known as **row-major order**.

Figure 5.14.1: Using a two-dimensional array: A driving distance between cities example.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

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

/* Direct driving distances between cities, in miles */
/* 0: Boston 1: Chicago 2: Los Angeles */

int main() {
    int cityA; // Starting city
    int cityB; // Destination city
    int drivingDistances[3][3]; // Driving distances

    // Initialize distances array
    drivingDistances[0][0] = 0;
    drivingDistances[0][1] = 960; // Boston-Chicago
    drivingDistances[0][2] = 2960; // Boston-Los Angeles
    drivingDistances[1][0] = 960; // Chicago-Boston
    drivingDistances[1][1] = 0;
    drivingDistances[1][2] = 2011; // Chicago-Los Angeles
    drivingDistances[2][0] = 2960; // Los Angeles-Boston
    drivingDistances[2][1] = 2011; // Los Angeles-Chicago
    drivingDistances[2][2] = 0;

    cout << "0: Boston 1: Chicago 2: Los Angeles" <<
endl;

    cout << "Enter city pair (Ex: 1 2) -- ";
    cin >> cityA;
    cin >> cityB;

    if ((cityA >= 0) && (cityA <= 2) && (cityB >= 0) &&
(cityB <= 2)) {
        cout << "Distance: " << drivingDistances[cityA]
[cityB];
        cout << " miles." << endl;
    }

    return 0;
}
```

```
0: Boston 1: Chicago 2: Los
Angeles
Enter city pair (Ex: 1 2) --
1 2
Distance: 2011 miles.

...
0: Boston 1: Chicago 2: Los
Angeles
Enter city pair (Ex: 1 2) --
2 0
xiang zhao
Distance: 2960 miles.

...
0: Boston 1: Chicago 2: Los
Angeles
Enter city pair (Ex: 1 2) --
1 1
Distance: 0 miles.
```

A programmer can initialize a two-dimensional array's elements during declaration using nested braces, as below. Multiple lines make the rows and columns more visible.

### Construct 5.14.1: Initializing a two-dimensional array during declaration.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

```
// Initializing a 2D array
int numVals[2][3] = { {22, 44, 66}, {97, 98, 99} };

// Use multiple lines to make rows more visible
int numVals[2][3] = {
    {22, 44, 66}, // Row 0
    {97, 98, 99} // Row 1
};
```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

Arrays of three or more dimensions can also be declared, as in `int myArray[2][3][5]`, which declares a total of  $2 \times 3 \times 5$  or 30 elements. Note the rapid growth in size -- an array declared as `int myArray[100][100][5][3]` would have  $100 \times 100 \times 5 \times 3$  or 150,000 elements. A programmer should make sure not to unnecessarily occupy available memory with a large array.

**PARTICIPATION ACTIVITY**

## 5.14.2: Two-dimensional arrays.



- 1) Declare a two dimensional array of integers named `dataVals` with 4 rows and 7 columns.

**Check****Show answer**

- 2) How many total elements are in an array with 4 rows and 7 columns?

**Check****Show answer**

- 3) How many elements are in the array declared as: `char streetNames[20][50];`

**Check****Show answer**

- 4) Write a statement that assigns 99 into the fifth row, third column of array `numVals`. Note: the first row/column is at index 0, not 1.

**Check****Show answer**

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



**CHALLENGE  
ACTIVITY****5.14.1: Find 2D array max and min.**

Find the maximum value and minimum value in milesTracker. Assign the maximum value to maxMiles, and the minimum value to minMiles. Sample output for the given program:

```
Min miles: -10
Max miles: 40
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

(Notes)

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     const int NUM_ROWS = 2;
6     const int NUM_COLS = 2;
7     int milesTracker[NUM_ROWS][NUM_COLS];
8     int i;
9     int j;
10    int maxMiles = 0; // Assign with first element in milesTracker before loop
11    int minMiles = 0; // Assign with first element in milesTracker before loop
12    int value;
13
14    for (i = 0; i < NUM_ROWS; i++){
15        for (j = 0; j < NUM_COLS; j++){
16            cin >> value;
17            milesTracker[i][j] = value;
18        }
19    }
20}
```

**Run**

View your last submission ▾

## 5.15 Char arrays / C strings

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

A programmer can use an array to store a sequence of characters, known as a **string**. Char arrays were the only kinds of strings in C++'s predecessor language C, and thus are sometimes called **C strings** to distinguish them from C++'s string type. An example is:

`char movieTitle[20] = "Star Wars";` Because a string can be shorter than the character array, a string in a char array must end with a special character known as a **null character**, written as '\0'. Given a string literal like "Star Wars", the compiler automatically appends a null character.

**PARTICIPATION ACTIVITY**

5.15.1: A char array declaration and initialization with null-terminated string.

**Animation captions:**

1. The character array must be large enough to include the null character. The compiler automatically inserts the null character to indicate the end of a string.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

A char array of size 20 can store strings of lengths 0 to 19. The longest string is 19, not 20, since the null character must be stored.

If a char array is initialized when declared, then the char array's size may be omitted, as in

`char userName[ ] = "Hellen";`. The compiler determines the size from the string literal, in this case 6 + 1 (for the null character), or 7.

An array of characters ending with a null character is known as a **null-terminated string**.

Output streams automatically handle null-terminated strings, printing each character until reaching the null character that ends the string.

Figure 5.15.1: Printing stops when reaching the null character at each string's end.

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

int main() {
    char cityName[20] = "Forest Lake"; // Compiler appends null char

    // In each cout, printing stops when reaching null char
    cout << "City:" << endl;           // Compiler appends null char to "City:"
    cout << cityName << endl;

    return 0;
}
```

City:  
Forest Lake

**PARTICIPATION ACTIVITY**

5.15.2: Char array strings.



©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

Indicate whether the array declaration and initialization are appropriate.

1) `char firstName[10] = "Henry";`



- True
- False

2)



```
char lastName[10] = "Michelson";
```

- True
- False

3) `char favoriteMuseum[10] = "Smithsonian";`

- True
- False

4) Printing catBreed will print 19 characters.

```
char catBreed[20] = "Persian";
```

- True
- False

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

After a string is declared, a programmer may not later assign the string as in

`movieTitle = "Indiana Jones";`. That statement tries to assign a value to the char array variable itself, rather than copying each character from the string on the right into the array on the left. Functions exist to copy strings, such as `strcpy()`, discussed elsewhere.

A programmer can traverse a string using a loop that stops when reaching the null character.

A common error is to loop for the string's array size rather than stopping at the null character. Such looping visits unused array elements beyond the null character. An even worse common error is to loop beyond the last valid element, which visits memory locations that are not part of the array. These errors are illustrated below. Notice the strange characters that are output as the contents of other memory locations are printed out; the program may also crash.

Figure 5.15.2: Traversing a C string.

```
Enter string (<20 chars):  
test@gmail.com
```

```
test@gmail.com  
Found '@'.
```

```
"test@gmail.com6789"03/11/21 19:10 488201  
"test@gmail.com6789P!"xiang zhao
```

BAYLORCSI14301440Spring2021

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

int main() {
    char userStr[20] = "1234567890123456789"; // Input string
    int i;

    // Prompt user for string input
    cout << "Enter string (<20 chars): ";
    cin >> userStr;

    // Print string
    cout << endl << userStr << endl;

    // Look for '@'
    for (i = 0; userStr[i] != '\0'; ++i) {
        if (userStr[i] == '@') {
            cout << "Found '@'." << endl;
        }
    }
    cout << endl;

    // The following is an ERROR.
    // May print chars it shouldn't.
    // Problem: doesn't stop at null char.
    cout << "\""; // Print opening "
    for (i = 0; i < 20; ++i) { // Print each char
        cout << userStr[i];
    }
    cout << "\"" << endl; // Print closing "

    // The following is an even WORSE ERROR.
    // Accesses beyond valid index range.
    // Program may crash.
    cout << "\""; // Print opening "
    for (i = 0; i < 30; ++i) {
        cout << userStr[i];
    }
    cout << "\"" << endl; // Print closing "

    return 0;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

The above output is machine and compiler dependent. Also, some values aren't printable so don't appear in the output.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

#### PARTICIPATION ACTIVITY

#### 5.15.3: C string errors.



Given the following char array declaration, which of the following code snippets are bad?

`char userText[10] = "Car";`

1)



```
for (i = 0; userText != '\0'; ++i) {
    // Print userText[i]
}
```

 OK Bad

2) **for** (*i* = 0; *userText*[*i*] != '\0';
 ++*i*) {
 // Print *userText*[*i*]
}

 OK Bad

3) **for** (*i* = 0; *i* < 10; ++*i*) {
 // Print *userText*[*i*]
}

 OK Bad

4) **for** (*i* = 0; *i* < 4; ++*i*) {
 // Print *userText*[*i*]
}

 OK Bad

5) *userText* = "Bus";

 OK Bad

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021



Yet another common error with C strings is for the program user to enter a string larger than the character array. That may cause the input statement to write to memory locations outside the array's locations, which may corrupt other parts of program or data, and typically causes the program to crash.

### zyDE 5.15.1: Reading in a string too large for a C string.

Run the program, which simply reads an input string and prints it one character at a time. Then, lengthen the input string beyond 10 characters, and run again. The program *might* if the extra memory locations being assigned don't matter. Try larger and larger strings, see if the program fails (be sure to scroll to the bottom of the output to look for erroneous output or an error message).

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

[Load default template...](#)

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     char userStr[10]; // Input string
6     int i;
7
8     // Prompt user for string input
9     cout << "Enter string (<10 chars):";
10    cin >> userStr;
11
12    // Print 1 char at a time
13    cout << endl;
14    for (i = 0; userStr[i] != '\0'; ++i)
15        cout << userStr[i] << endl;
16    }
17    cout << endl;
18

```

Hello

Run

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

C string usage is fraught with common errors. C++ introduced its own string type, as in `string myString;` and accessible after `#include <string>`, to reduce those errors and increase programmer convenience. C strings are still used in some legacy code and are thus good to learn. C++ provides common functions for handling C strings, which can be used by including the following: `#include <cstring>`.

The following program is for illustration, showing how a string is made up of individual character elements followed by a null character. Normally a programmer would not create a string that way.

Figure 5.15.3: A C string is an array of characters, ending with the null character.

```

#include <iostream>
using namespace std;

int main() {
    char nameArr[5];

    nameArr[0] = 'A';
    nameArr[1] = 'l';
    nameArr[2] = 'a';
    nameArr[3] = 'n';
    nameArr[4] = '\0'; // Null character

    cout << nameArr << endl;

    nameArr[4] = '!'; // Oops, overwrote null char
    cout << nameArr << endl; // *Might* still work

    return 0;
}

```

Alan  
Alan!

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

When printing a string stored within a character array, each character within the array will be printed until the null character is reached. If the null character is omitted, the program would print whatever values are found in memory after the array, until a null character happens to be encountered. Omitting the null character is a serious logical error.

It just so happens that the null character '\0' has an ASCII encoding of 0. Many compilers initialize memory to 0s. As such, omitting the '\0' in the above program would not always cause erroneous execution. Like a nail in the road, that bug in your code is just waiting to wreak havoc.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021



#### PARTICIPATION ACTIVITY

#### 5.15.4: C string without null character.

Given:

```
char userText[10];
userText[0] = 'c';
userText[1] = 'a';
userText[2] = 'r';
userText[3] = '\0';
...
userText[3] = 's';
```

- 1) The first four characters in userText are now: Cars.

- True
- False

- 2) The compiler generates an error, because element 3 is the null character and can't be overwritten.

- True
- False

- 3) Printing userText should work fine because the new string is 4 characters, which is still much less than the array size of 10.

- True
- False

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

## 5.16 Multiple arrays

Programmers commonly use multiple same-sized arrays to store related lists. The program below maintains a list of letter weights in ounces, and another list indicating the corresponding postage cost

for first class mail (usps.com).

The `if (userLetterWeight <= letterWeights[i])` statement compares the user-entered letter weight with the current element in the letterWeights array. If the entered weight is less than or equal to the current element in the letterWeights array, the program prints the element in postageCosts at the same index.

The loop's expression `(i < NUM_ELEMENTS) && (!foundWeight)` depends on the value of the variable foundWeight. This expression prevents the loop from iterating through the entire array once the correct letter weight has been found. Omitting the check for found from the loop expression would result in an incorrect output; the program would incorrectly print the postage cost for all letter weights greater than the user's letter weight.

Figure 5.16.1: Multiple array example: Letter postage cost program.

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

int main () {
    const int NUM_ELEMENTS = 14;
    // Weights in ounces
    double letterWeights[NUM_ELEMENTS] = {1.0, 2.0, 3.0, 3.5, 4.0, 5.0, 6.0,
                                           7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0};
    // Costs in cents (usps.com 2017)
    int postageCosts[NUM_ELEMENTS] = {49, 70, 91, 112, 161, 182, 203,
                                      224, 245, 266, 287, 308, 329, 350};
    double userLetterWeight;
    bool foundWeight;
    int i;

    // Prompt user to enter letter weight
    cout << "Enter letter weight (in ounces): ";
    cin >> userLetterWeight;

    // Postage costs is based on smallest letter weight greater than
    // or equal to mailing letter weight
    foundWeight = false;

    for (i = 0; (i < NUM_ELEMENTS) && (!foundWeight); ++i) {
        if (userLetterWeight <= letterWeights[i]) {
            foundWeight = true;
            cout << "Postage for USPS first class mail is ";
            cout << postageCosts[i] << " cents" << endl;
        }
    }

    if( !foundWeight ) {
        cout << "Letter is too heavy for USPS first class mail." << endl;
    }
}

return 0;
}
```

```
Enter letter weight (in ounces): 3
Postage for USPS first class mail is 91 cents

...
Enter letter weight (in ounces): 9.5
Postage for USPS first class mail is 287 cents

...
Enter letter weight (in ounces): 15
Letter is too heavy for USPS first class mail.
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**PARTICIPATION ACTIVITY**

5.16.1: Multiple arrays in the above postage cost program.

- 1) letterWeights[0] is 1, meaning the first element of letterWeights and postageCosts correspond to a weight of 1 ounce.

- True
- False

- 2) postageCosts[2] represents the cost for a weight of 2 ounces.

- True
- False

- 3) The program fails to provide a cost for a weight of 7.5.

- True
- False

### zyDE 5.16.1: Postage calculation with negative weight error message:

Improve the program by also outputting "The next higher weight is \_\_ with a cost of \_\_ cents".

Load default template...

```
1 #include <iostream>
2 using namespace std;
```

3

Run

```
3 int main () {
4     const int NUM_ELEMENTS = 14;
5     // Weights in ounces
6     double letterWeights[NUM_ELEMENTS] =
7         .....
8     // Costs in cents (usps.com 2017)
9     int postageCosts[NUM_ELEMENTS] = {49
10        .....
11        22
12     double userLetterWeight;
13     bool foundWeight;
14     int i;
15
16     // Prompt user to enter letter weight
17     cout << "Enter letter weight (in ou
18 }
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**PARTICIPATION ACTIVITY**

## 5.16.2: Multiple arrays.



- 1) Using two separate statements, declare two related integer arrays named seatPosition and testScore (in that order) each with 130 elements.

**Check****Show answer**

- 2) How many total elements are stored within the two arrays int familyAges[50] and double familyHeights[50]?

**Check****Show answer****CHALLENGE ACTIVITY**

## 5.16.1: Multiple arrays.



©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

## 5.17 String library functions

C++ provides functions for working with C strings, presented in the ***cstring*** library. To use those functions, the programmer starts with: `#include <cstring>`.

Some C string functions for *modifying* strings are summarized below.

Table 5.17.1: Some C string modification functions.

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

Given:

```
char orgName[100] = "United Nations";
char userText[20] = "UNICEF";
char targetText[10];
```

|                  |  |  |
|------------------|--|--|
| <b>strcpy()</b>  | <pre>strcpy(destStr, sourceStr)</pre> <p>Copies sourceStr (up to and including null character) to destStr.</p>   | <pre>strcpy(targetText, userText); // Copies "UNICEF" + null char  // to targetText strcpy(targetText, orgName); // Error: "United Nations"  // has &gt; 10 chars targetText = orgName; // Error: Strings can't be  // copied this way</pre> |
| <b>strncpy()</b> | <pre>strncpy(destStr, sourceStr, numChars)</pre> <p>Copies up to numChars characters.</p>  | <pre>strncpy(orgName, userText, 6); // orgName is "UNICEF Nations"</pre>   |
| <b>strcat()</b>  | <pre>strcat(destStr, sourceStr)</pre> <p>Copies sourceStr (up to and including null character) to end of destStr (starting at destStr's null character).</p> | <pre>strcat(orgName, userText); // orgName is "United NationsUNICEF"</pre>   |
| <b>strncat()</b> | <pre>strncat(destStr, sourceStr, numChars)</pre> <p>Copies up to numChars characters to destStr's end, then appends null character.</p>                      | <pre>strcpy(targetText, "abc"); // targetText is "abc" strncat(targetText, "123456789", 3); // targetText is "abc123" targetText is 01"abc123"ing2021</pre>  |

For strcpy(), a common error is to copy a source string that is too large, causing an out-of-range access in the destination string. Another common error is to call strcpy with the source string first rather than the destination string, which copies in the wrong direction.

Note that string assignment, as in `targetText = orgName`, does not copy the string and should not be used. The exception is during initialization, as in `char userText[20] = "UNICEF";`, for which the compiler copies the string literal's characters into the array.

**PARTICIPATION ACTIVITY**

## 5.17.1: String modification functions.



Given: `char userStr[5];`

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

Do not type quotes in your answers.  
If the function call is incorrect, causes an out-of-range access, or the resulting string does not end with a null character, type Error.

- 1) What is `userStr` after: `strcpy(userStr, "Bye");`

**Check****Show answer**

- 2) If `userStr` is initially "Hi", what is `userStr` after: `strcpy(userStr, "Bye");`

**Check****Show answer**

- 3) What is `userStr` after: `strcpy(userStr, "Goodbye");`

**Check****Show answer**

- 4) If `userStr` is initially "Hi!!", what is `userStr` after: `strncpy(userStr, "Bye", 3);`

**Check****Show answer**

- 5) What is `userStr` after: `strncpy(userStr, "Goodbye", 4);`

**Check****Show answer**

- 6) If userStr is initially "Hi", what is userStr  
after: strcat(userStr, '!');

[Show answer](#)

- 7) If userStr is initially "Hi", what is userStr  
after: strcat(userStr, "!");

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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

- 8) If userStr is initially "Hi", what is userStr  
after: strncat(userStr, "?!\$#@%", 2);

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

Several C string functions that get *information* about strings are summarized below.

Table 5.17.2: Some C string information functions.

Given:

```
char orgName[100] = "United Nations";
char userText[20] = "UNICEF";
char targetText[10];
```

|                 |  |   |
|-----------------|--|---|
| <b>strchr()</b> | <b>strchr(sourceStr, searchChar)</b><br><br>Returns NULL if searchChar does not exist in sourceStr. (Else, returns address of first occurrence, discussed elsewhere).<br>NULL is defined in the cstring library. | <b>if (strchr(orgName, 'U') != NULL)</b><br><b>{ // 'U' exists in orgName?</b><br><b>... // 'U' exists in "United Nations", branch taken</b><br><b>}</b><br><b>if (strchr(orgName, 'u') != NULL)</b><br><b>{ // 'u' exists in orgName?</b><br><b>... // 'u' doesn't exist (case matters), branch not taken</b><br><b>}</b><br>xiang zhao<br>BAYLORCSI14301440Spring2021 |
| <b>strlen()</b> | <b>size_t strlen(sourceStr)</b><br><br>Returns number of characters in sourceStr up to, but not including, first null character. size_t is integer type.   | x = <b>strlen(orgName);</b> // Assigns 14 to x<br>x = <b>strlen(userText);</b> // Assigns 6 to x<br>x = <b>strlen(targetText);</b> // Error: targetText may lack null char  |

**strcmp()**

```
int strcmp(str1, str2)
```

Returns 0 if str1 and str2 are equal, non-zero if they differ.

```
if (strcmp(orgName, "United
Nations") == 0) {
    ... // Equal, branch taken
}
if (strcmp(orgName, userText) != 0) {
    ... // Not equal, branch not
taken
}
```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

strcmp() is usually used to compare for equality, returning 0 if the strings are the same length and have identical characters. A common error is to use == when comparing C strings, which does not work. str1 == str2 compares the strings' addresses, not their contents. Because those addresses will usually be different, str1 == str2 will evaluate to 0. This is not a syntax error, but clearly a logic error. Another common error is to forget to compare the result of strcmp with 0, as in

`if (strcmp(str1, str2)) {...}.` The code is not a syntax error, but is a logic error because the if condition will be false (0) when the strings are equal. The correct condition would instead be `if (strcmp(str1, str2) == 0) {...}.` Although strcmp returns 0, a good practice is to avoid using `if (!strcmp(str1,str2)) {...}` because that 0 does not represent "false" but rather is encoding a particular situation.

strcmp(str1, str2) returns a negative number if str1 is less than str2, and a positive number if str1 is greater than str2. Evaluation first compares the character pair at element 0, then at element 1, etc., returning as soon as a pair differs.

**PARTICIPATION ACTIVITY**

5.17.2: String comparison.

**Animation captions:**

1. Each comparison uses ASCII values.
2. Values at indexes 0-4 are the same for both student\_name and teacher\_name.
3. 'J' is greater than 'A', so student\_name is greater than teacher\_name.

strlen is often used to iterate through each string character in a loop.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

Figure 5.17.1: Iterating through a C string using strlen.

|                     |
|---------------------|
| Before: Alan Turing |
| After: Alan_Turing  |

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

int main() {
    char userName[15] = "Alan Turing";
    int i;

    cout << "Before: " << userName << endl;

    for (i = 0; i < strlen(userName); ++i) {
        if (userName[i] == ' ') {
            userName[i] = '_';
        }
    }
    cout << "After: " << userName << endl;

    return 0;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**PARTICIPATION ACTIVITY**

5.17.3: Some C string library functions.



### Animation captions:

1. The `strlen()` function returns the number of characters in `str1` up to, but not including, the first null character.
2. The `strcpy()` function copies `str2` to `str1`, up to and including `str2`'s null character.
3. The `strcat()` function starts at `str1`'s null character, then copies `str2` to `str1`, up to and including `str2`'s null character.
4. The `strchr()` function returns the address of the first occurrence of 'w' in `str1`. If 'w' does not exist in `str1`, then the `strchr()` function returns `NULL`.

**PARTICIPATION ACTIVITY**

5.17.4: String information functions.



Given:

```
char str1[10] = "Earth";
char str2[20] = "Earthlings";
char str3[15] = "Mars";
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

Answer the following questions. If appropriate, type: Error

- 1) What does `strlen(str3)` return?



**Check****Show answer**

- 2) Is the branch taken? (Yes/No/Error)



```
if (strchr(str1, '@') != NULL) {  
    // Print "Found @"  
}
```

**Check****Show answer**

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

- 3) Is the branch taken? (Yes/No/Error)



```
if (strchr(str1, 'E') != NULL) {  
    // Print "Found E"  
}
```

**Check****Show answer**

- 4) Is the branch taken? (Yes/No/Error)



```
if (strchr(str2, "Earth") != NULL)  
{  
    // Print "Found Earth"  
}
```

**Check****Show answer**

- 5) Is the branch taken? (Yes/No/Error)



```
if (strcmp(str1, str2) == 0) {  
    // Print "strings are equal"  
}
```

**Check****Show answer**

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

- 6) Is the branch taken? (Yes/No/Error)



```
if (str1 == str3) {  
    // Print "strings are equal"  
}
```

**Check****Show answer**

- 7) Finish the code to take the branch if str1 and str3 are equal.



```
if (strcmp(str1, str3)  ) {
    // Strings are equal
}
```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

**Check****Show answer**

Exploring further:

- More C string functions from [cplusplus.com](http://www.cplusplus.com)

## 5.18 Char library functions: ctype

C++ provides common functions for working with characters, presented in the `cctype` library, the `ctype` short for "character type", and the first c indicating the library is a C language standard library. To use those functions, the programmer adds the following at the top of a file: `#include <cctype>`

Commonly-used `cctype` functions are summarized below; a complete reference is found at <http://www.cplusplus.com/reference/cctype/>.

### Character checking functions

The following functions check whether a character is of a given category, returning either false (0) or true (non-zero).

Table 5.18.1: Functions that check whether a character is of a given category.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

The examples below assume the following string declaration.

```
char myString[30] = "Hey9! Go";
```

**isalpha(c)** -- Returns true if c is alphabetic: a-z or A-Z

|   |  |
|---|--|
| Z.  | <pre>isalpha('A');           // Returns true isalpha(myString[0]);   // Returns true because 'H' is alphabetic isalpha(myString[3]);   // Returns false because '9' is not alphabetic</pre>  |
| <b>isdigit(c)</b> -- Returns true if c is a numeric digit: 0-9.   | <pre>isdigit(myString[3]);    // Returns true because '9' is numeric isdigit(myString[4]);    // Returns false because ! is not numeric</pre>  |
| <b>isalnum(c)</b> -- Returns true if c is alphabetic or a numeric digit. Thus, returns true if either isalpha or isdigit would return true. | <pre>isalnum('A');           // Returns true isalnum(myString[3]);   // Returns true because '9' is numeric</pre>  |
| <b>isspace(c)</b> -- Returns true if character c is a whitespace.   | <pre>isspace(myString[5]);   // Returns true because that character is a space ' '. isspace(myString[0]);   // Returns false because 'H' is not whitespace.</pre>  |
| <b>islower(c)</b> -- Returns true if character c is a lowercase letter a-z.   | <pre>islower(myString[0]);   // Returns false because 'H' is not lowercase. islower(myString[1]);   // Returns true because 'e' is lowercase. islower(myString[3]);   // Returns false because '9' is not a lowercase letter.</pre>              |
| <b>isupper(c)</b> -- Returns true if character c is an uppercase letter A-Z.  | <pre>isupper(myString[0]);   // Returns true because 'H' is uppercase. isupper(myString[1]);   // Returns false because 'e' is not uppercase. isupper(myString[3]);   // Returns false because '9' is not an uppercase letter.</pre>             |
| <b>isblank(c)</b> -- Returns true if character c is a blank character. Blank characters include spaces and tabs.                            | <pre>isblank(myString[5]);   // Returns true because that character is a space ' '. isblank(myString[0]);   // Returns false because 'H' is not blank.</pre>   |
| <b>isxdigit(c)</b> -- Returns true if c is a hexadecimal digit: 0-9, a-f, A-F.  | <pre>isxdigit(myString[3]); // Returns true because '9' is a hexadecimal digit. isxdigit(myString[1]); // Returns true because 'e' is a hexadecimal digit. isxdigit(myString[6]); // Returns false because 'G' is not a hexadecimal digit.</pre> |
| <b>ispunct(c)</b> -- Returns true if c is a punctuation   |  |

|   |  |
|---|--|
| <p>character. Punctuation characters include:<br/>!"#\$%&amp;'()*+,-./;=&gt;?<br/>@[{}]^_`{}`~</p>  | <pre>ispunct(myString[4]); // Returns true because '!' is a punctuation character. ispunct(myString[6]); // Returns false because 'G' is not a punctuation character.</pre>  |
| <p><b>isprint(c)</b> -- Returns true if c is a printable character. Printable characters include alphanumeric, punctuation, and space characters.</p> | <pre>isprint(myString[0]); // Returns true because 'H' is a alphabetic. isprint(myString[4]); // Returns true because '!' is a punctuation. isprint(myString[5]); // Returns true because that character is a space ' '. isprint('\0'); // Returns false because the null character is not printable</pre> |
| <p><b>iscntrl(c)</b> -- Returns true if c is a control character. Control characters are all characters that are not printable.</p>                   | <pre>iscntrl(myString[0]); // Returns false because 'H' is a not a control character iscntrl(myString[5]); // Returns false because space is a not a control character iscntrl('\0'); // Returns true because the null character is a control character</pre>  |

## Character conversion functions

The following functions return a character representing a converted version of the input character.

Table 5.18.2: Functions that convert a character is of a given category.

The examples below assume the following string declaration.

```
char myString[30] = "Hey9! Go";
```

|  |   |
|--|---|
| <p><b>toupper(c)</b> -- If c is a lowercase alphabetic character (a-z), returns the uppercase version (A-Z). If c is not a lowercase alphabetic character, just returns c.</p> | <pre>letter = toupper(myString[0]); // Returns 'H' (no change) letter = toupper(myString[1]); // Returns 'E' ('e' converted to 'E') letter = toupper(myString[3]); // Returns '9' (no change) letter = toupper(myString[5]); // Returns ' ' (no change)</pre> |
| <p><b>tolower(c)</b> -- If c is an uppercase alphabetic character (A-Z), returns the lowercase version (a-z). If c is not an</p>   | <pre>letter = tolower(myString[0]); // Returns 'h' ('H' converted to 'h') letter = tolower(myString[1]); // Returns 'e' (no change) letter = tolower(myString[3]); // Returns '9' (no change) letter = tolower(myString[5]); // Returns ' ' (no change)</pre> |

uppercase alphabetic character, just returns c.

The following example illustrates some of the ctype functions.

Figure 5.18.1: Use of some functions in ctype.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

```
Enter string (<30 chars):  
ABC123$!def  
Original: ABC123$!def  
isalpha: YYYNNNNNNYYY  
isdigit: NNNYYYYNNNNN  
isupper: YYYNNNNNNNNN  
After toupper: ABC123$!DEF
```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

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

int main() {
    const int MAX_LEN = 30;           // Max string length
    char userStr[MAX_LEN];          // User defined
    string
    int i;

    // Prompt user to enter string
    cout << "Enter string (<" 
    << MAX_LEN << " chars): ";
    cin >> userStr;

    cout << "Original: " << userStr << endl;

    cout << "isalpha: ";
    for (i = 0; userStr[i] != '\0'; ++i) {
        if (isalpha(userStr[i])) {
            cout << "Y";
        }
        else {
            cout << "N";
        }
    }
    cout << endl;

    cout << "isdigit: ";
    for (i = 0; userStr[i] != '\0'; ++i) {
        if (isdigit(userStr[i])) {
            cout << "Y";
        }
        else {
            cout << "N";
        }
    }
    cout << endl;

    cout << "isupper: ";
    for (i = 0; userStr[i] != '\0'; ++i) {
        if (isupper(userStr[i])) {
            cout << "Y";
        }
        else {
            cout << "N";
        }
    }
    cout << endl;

    for (i = 0; userStr[i] != '\0'; ++i) {
        userStr[i] = toupper(userStr[i]);
    }
    cout << "After toupper: " << userStr << endl;

    return 0;
}
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

To compare two strings without paying attention to case, one technique is to first convert (a copy of) each string to lowercase (using a loop, discussed elsewhere) and then comparing.



Enter the value to which each function evaluates using 1 for true, 0 for false for boolean functions.

Assume str is "Hi 321!".

1) `isalpha(str[0])`

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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

2) `isdigit(str[4])`

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

3) `isalnum(str[2])`

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

4) `isspace(str[2])`

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

5) `islower(str[6])`

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

6) `tolower(str[0])`

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

7) `tolower(str[1])`

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

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

# 5.19 C++ example: Annual salary tax rate calculation with vectors

zyDE 5.19.1: Various tax rates.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

Vectors are useful to process tabular information. Income taxes are based on annual salary usually with a tiered approach. Below is an example of a simple tax table:

| Annual Salary         | Tax Rate |
|-----------------------|----------|
| 0 to 20000            | 10%      |
| Above 20000 to 50000  | 20%      |
| Above 50000 to 100000 | 30%      |
| Above 100000          | 40%      |

The below program uses a vector salaryBase to hold the cutoffs for each salary level and a parallel vector taxBase that has the corresponding tax rate.

1. Run the program and enter annual salaries of 40000 and 60000, then enter 0.
2. Modify the program to use two parallel vectors named annualSalaries and taxesToPay, each with 10 elements. Vectors annualSalaries holds up to 10 annual salaries entered by the user. Vector taxesToPay holds up to 10 corresponding amounts of taxes to pay for those annual salaries. Print the total annual salaries and taxes to pay after all input has been processed.
3. Run the program again with the same annual salary numbers as above.

The following program calculates the tax rate and tax to pay based on annual income.

Load default template

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     const int MAX_ELEMENTS = 10;
7     int annualSalary;
8     double taxRate;
9     int taxToPay;
10    int numSalaries;
11    bool keepLooking;
12    unsigned int i;
13    vector<int> salaryBase(5);

```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

```
14     vector<double> taxBase(5);
15     // FIXME: Declare annualSalaries and taxesToPay vectors to hold 10 elements
16     // FIXME: Use the constant MAX_ELEMENTS to declare the vectors
17
18     salaryBase.at(0) = 0;
```

```
40000 60000 0
```

**Run**

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

## zyDE 5.19.2: Various tax rates (solution).

A solution to the problem follows.

**Load default templ**

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     const int MAX_ELEMENTS = 10;
7     int annualSalary;
8     double taxRate;
9     int taxToPay;
10    int totalSalaries;
11    int totalTaxes;
12    int numSalaries;
13    bool keepLooking;
14    unsigned int i;
15    int j;
16
17    vector<int> salaryBase(5);
18    vector<double> taxBase(5);
```

```
40000 60000 0
```

©zyBooks 03/11/21 19:10 488201  
xiang zhao  
BAYLORCSI14301440Spring2021

**Run**

# 5.20 C++ example: Domain name validation with vectors

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

zyDE 5.20.1: Validate domain names with vectors.

Vectors are useful to process lists.

A **top-level domain** (TLD) name is the last part of an Internet domain name like .com in example.com. A **core generic top-level domain** (core gTLD) is a TLD that is either .com, .org, or .info. A **restricted top-level domain** is a TLD that is either .biz, .name, or .pro. A **second-level domain** is a single name that precedes a TLD as in apple in apple.com.

The following program repeatedly prompts for a domain name, and indicates whether the domain name consists of a second-level domain followed by a core gTLD. Valid core gTLDs are stored in a vector. For this program, a valid domain name must contain only one per second-level domain such as apple.com, but not support.apple.com. The program ends when the user enters a blank line.

1. Run the program and enter domain names to validate.
2. Extend the program to also recognize restricted TLDs using a vector, and statement validate against that vector. The program should also report whether the TLD is a core gTLD or a restricted gTLD. Run the program again.

[Load default template](#)

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     // Define the list of valid core gTLDs
7     const int NUM_ELEMENTS = 4;
8     vector<string> validCoreGtld(NUM_ELEMENTS);
9     // FIXME: Declare a vector named validRestrictedGtld that has the names
10    //         of the restricted domains, .biz, .name, and .pro
11    string inputName;
12    string searchName;
13    string theGtld;
14    bool isValidDomainName;
15    bool isCoreGtld;
16    bool isRestrictedGtld;
17    int periodCounter;
18    int periodPosition;

```

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

apple.com  
APPLE.com  
apple.comm

Run

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

## zyDE 5.20.2: Validate domain names with vectors (solution).

A solution to the problem follows.

Load default templ

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     // Define the list of valid core gTLDs
7     const int NUM_ELEMENTS_CORE = 4;
8     vector<string> validCoreGtld(NUM_ELEMENTS_CORE);
9     const int NUM_ELEMENTS_RSTR = 3;
10    vector<string> validRestrictedGtld(NUM_ELEMENTS_RSTR);
11    string inputName;
12    string searchName;
13    string theGtld;
14    bool isValidDomainName;
15    bool isCoreGtld;
16    bool isRestrictedGtld;
17    int periodCounter;
18    int periodPosition;
```

apple.com  
APPLE.com  
apple.comm

Run

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

## 5.21 LAB: Output numbers in reverse

Write a program that reads a list of integers, and outputs those integers in reverse. The input begins with an integer indicating the number of integers that follow. For coding simplicity, follow each output integer by a comma, including the last one.

Ex: If the input is:

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

```
5 2 4 6 8 10
```

the output is:

```
10,8,6,4,2,
```

To achieve the above, first read the integers into a vector. Then output the vector in reverse.

### LAB ACTIVITY

#### 5.21.1: LAB: Output numbers in reverse

10 / 10



main.cpp

1 Loading latest submission...

Develop mode

Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

```
6 2 3 3 4 5 6 6
```

**Run program**

Input (from above)

**main.cpp**  
(Your program)

Outp

Program output displayed here

Signature of your work

[What is this?](#)

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

Retrieving signature

## 5.22 LAB: Middle item

Given a sorted list of integers, output the middle integer. A negative number indicates the end of the input (the negative number is not a part of the sorted list). Assume the number of integers is always odd.

Ex: If the input is:

2 3 4 8 11 -1

the output is:

Middle item: 4

The maximum number of inputs for any test case should not exceed 9. If exceeded, output "Too many numbers".

Hint: First read the data into a vector. Then, based on the number of items, find the middle item.

**LAB  
ACTIVITY**

5.22.1: LAB: Middle item

10 / 10



©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

1 Loading latest submission...

**Develop mode****Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

2 3 4 8 11 2 2 2 2 2 2 2 2 -1

**Run program**

Input (from above)

**main.cpp**  
(Your program)

Output

Program output displayed here

Signature of your work [What is this?](#)

Retrieving signature

## 5.23 LAB: Output values below an amount

Write a program that first gets a list of integers from input. The input begins with an integer indicating the number of integers that follow. Then, get the last value from the input, which indicates a threshold. Output all integers less than or equal to that last threshold value.

xiang zhao  
BAYLORCSI14301440Spring2021

Ex: If the input is:

5 50 60 140 200 75 100

the output is:

50, 60, 75,

The 5 indicates that there are five integers in the list, namely 50, 60, 140, 200, and 75. The 100 indicates that the program should output all integers less than or equal to 100, so the program outputs 50, 60, and 75.

For coding simplicity, follow every output value by a comma, including the last one.

Such functionality is common on sites like Amazon, where a user can filter results.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

**LAB  
ACTIVITY**

5.23.1: LAB: Output values below an amount

10 / 10



main.cpp

1 Loading latest submission...

**Develop mode**

**Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

**Run program**

Input (from above)



**main.cpp**  
(Your program)



Output

Program output displayed here

Signature of your work

[What is this?](#)



Retrieving signature

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021

©zyBooks 03/11/21 19:10 488201

xiang zhao

BAYLORCSI14301440Spring2021