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);
```

Feedback?

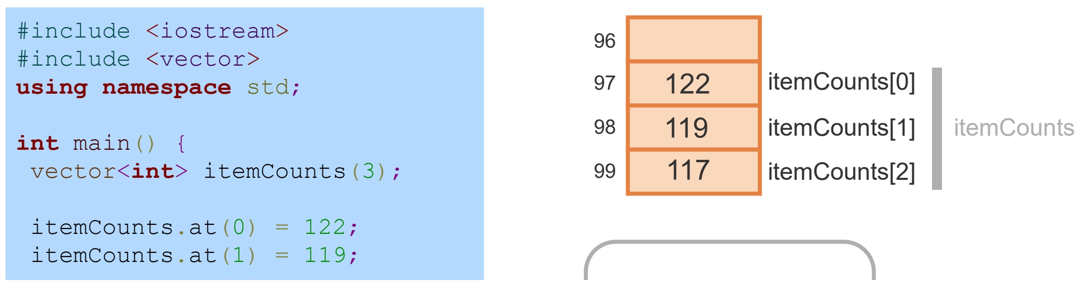
The statement above declares a vector with the specified number of elements, each element of the specified data type. The type of each vector element is specified within the angle brackets (<>). The number of vector elements is specified within parentheses following the vector name. Ex: `vector<int>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

1 2 2x speed



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

int main() {
 vector<int> itemCounts(3);

 itemCounts.at(0) = 122;
 itemCounts.at(1) = 119;
```

```
   itemCounts.at(2) = 117;

   cout << itemCounts.at(1);

   return 0;
}
```

```
119
```

An element is accessed with the at() function.
The number in parentheses is the index of the corresponding element.

Feedback?

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]

Correct

The declaration creates vector yearsList with 4 elements. The elements' indices will be 0, 1, 2, and 3.

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

- 1
- 1999
- [✓ 2012]

Correct

The element yearsList.at(1) is like an int variable. That element was assigned 2012.

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

- 2
- [✓ 2025]
- Invalid index

Correct

The element yearsList.at(2) is like an int variable. 2025 was earlier assigned to that element.

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

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

Correct

The valid indices start at 0, so the four elements are 0, 1, 2, and 3. Accessing yearsList.at(4) will cause an error.

✓

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

- ○ yearsList.at(1)
- [✓ yearsList.at(0)]

Correct

Vector elements are indexed starting with 0. That idea can be hard for new programmers to remember.

✓

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

- ○ 1999, 2012, 1999, 0
- ○ 2012, 2012, 2025, 0
- [✓ 2025, 2012, 2025, 0]

Correct

Each element is its own variable, and can be read and assigned just like any other variable.

✓

7) What is the index of the *last* element for the following vector: `vector<int> pricesList(100);`

- [✓ 99]
- ○ 100
- ○ 101

Correct

The 100 elements will have indices 0..99.

✓

Feedback?

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 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.

```
#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.

...

Enter N (1..5): 9

...

Enter N (1..5): 0

...

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

Feedback?

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.

Feedback?

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

Answer

`vector<int> myVals(10);`

The value inside the <> indicates the type of elements stored in the vector. The value inside the () defines the number of these elements.

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

Check Show answer

Answer

`x = myVals.at(8);`

Any element in a vector can be accessed by the .at() function as long as the element is in the vector's scope.

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

Check Show answer

Answer

`myVals.at(9) = 555;`

The last index in myVals is 9, so myVals.at(9) is the last vector element.

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

Answer

`myVals.at(currIndex) = 777;`

Check Show answer

The value held in currIndex is used as the vector index. A vector index may be any expression that evaluates to an integer value.

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

Check Show answer

Answer

```
tempVal = myVals.at(i + 1);
```

The expression `i + 1` is evaluated, and the result is used as the vector index.

Feedback?

Loops and vectors

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.

```
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
```

```
#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;
}
```

Feedback?

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) {`

Feedback?

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

Feedback?

Common error: Forgetting to include <vector>

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.

Jump to level 1

Type the program's output.

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

int main() {
   const int NUM_ELEMENTS = 3;
   vector<int> userVals(NUM_ELEMENTS);
   unsigned int i;

   userVals.at(0) = 3;
   userVals.at(1) = 4;
   userVals.at(2) = 9;

   userVals.at(1) = userVals.at(2);
   userVals.at(2) = userVals.at(1);

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

   return 0;
}
```

```
3
9
9
```

1 2 3 4

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

✓ The vector is initialized with 3, 4, and 9 in order. Then, the element at index 1 is assigned wi 2, and then the element at index 2 is assigned with the value of the element at index 1, which is 2.

Yours
```
3
9
9
```

Expected
```
3
9
9
```

Feedback?

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.

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

int main() {
   const int NUM_VALS = 5;
   vector<int> runTimes(NUM_VALS);
   unsigned int i;

   // Populate vector
   for (i = 0; i < runTimes.size(); ++i) {
      cin >> runTimes.at(i);
   }

   /* Your solution goes here  */
   for (i = 0; i < 3; ++i) {
      cout << runTimes.at(i) << endl;
   }
```

```
    return 0;
}

```

Run ✓ All tests passed

✓ Testing with inputs: 800 775 790 805 808

Your output

```
800
775
790
```

✓ Testing with inputs: 10 25 -10 20 25

Your output

```
10
25
-10
```

Feedback?

CHALLENGE ACTIVITY 5.2.3: Printing vector elements with a for loop. ✓

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.

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

int main() {
   const int NUM_VALS = 4;
   vector<int> courseGrades(NUM_VALS);
   int i;

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

   /* Your solution goes here  */
   for (i = 0; i < courseGrades.size(); ++i) {
      cout << courseGrades.at(i) << " ";
   }
   cout << endl;

   for (i =courseGrades.size()-1; i >=0; --i) {
      cout << courseGrades.at(i) << " ";
   }
   cout << endl;
```

Run

✓ All tests passed

✓ Testing with inputs: 7 9 11 10

Your output

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

✓ Testing with inputs: 70 99 85 60

Your output

```
70 99 85 60
60 85 99 70
```

Feedback?