

# Vectors

**Ahsan Ayub**

Ph.D. Student, Department of Computer Science

Graduate Research Assistant, CEROC

**CSC 1300: Introduction to Programming**

Monday, November 15, 2021

# Why use Vectors?

- Vectors are sequence containers representing arrays that can **change in size**.
- Just like arrays, vectors use contiguous memory locations for their elements.
- Internally, vectors use a dynamically allocated array to store their elements.
- However, compared to arrays, vectors consume more memory in exchange for the ability to manage storage and grow dynamically in an efficient way.

# Properties of Vectors

- **Sequence.** Elements in sequence containers are ordered in a strict linear sequence. Individual elements are accessed by their position in this sequence.
- **Dynamic Array.** Allows direct access to any element in the sequence.
- **Allocator-aware.** The container uses an allocator object to dynamically handle its storage needs.

# Introducing Vectors

Vectors are part of the *C++ Standard Template Library (STL)*. To use vectors, we need to include the **vector** header file in our program.

```
#include <vector>
```

# Declaration of Vectors

```
vector<data_type> vector_name
```

```
vector<int> myVector
```

```
vector<double> myVector
```

```
vector<char> myVector
```

```
vector<string> myVector
```

Note: We don't need to specify the size of the vector during the declaration.

# Vector Initialization

```
vector<int> myVector1 = {3, 1, 8}
```

```
vector<int> myVector2 {1, 5, 9}
```

# A Simple Program w/ Vector

```
1  #include <iostream>
2  // Library required to use vectors
3  #include <vector>
4  using namespace std;
5
6
7  int main()
8  {
9      // Initialize a vector
10     vector<int> myVector = {3, 1, 8};
11     // Iterate through each element
12     for(int i = 0; i < 3; i++)
13     {
14         cout << myVector[i] << endl;
15     }
16
17     return 0;
18 }
```

# A Simple Program w/ Vector

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
1  #include <iostream>
2  // Library required to use vectors
3  #include <vector>
4  using namespace std;
5
6
7  int main()
8  {
9      // Initialize a vector
10     vector<int> myVector = {3, 1, 8};
11     // Iterate through each element
12     for(int i = 0; i < 3; i++)
13     {
14         cout << myVector[i] << endl;
15     }
16
17     return 0;
18 }
```



# A Simple Program w/ Vector

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
3  
1  
8
```

```
1  #include <iostream>  
2  // Library required to use vectors  
3  #include <vector>  
4  using namespace std;  
5  
6  
7  int main()  
8  {  
9      // Initialize a vector  
10     vector<int> myVector = {3, 1, 8};  
11     // Iterate through each element  
12     for(int i = 0; i < 3; i++)  
13     {  
14         cout << myVector[i] << endl;  
15     }  
16  
17     return 0;  
18 }
```

# Vector Initialization

```
vector<int> myVector (5, 12);
```

↳ {12, 12, 12, 12, 12}

Note: This code creates an int vector with size 5 and initializes the vector with the value of 12.

# Add Element to a Vector: push\_back()

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
??
```

```
1  #include <iostream>
2  // Library required to use vectors
3  #include <vector>
4
5  using namespace std;
6
7  int main()
8  {
9      // Initialize a vector with some values
10     vector<int> myVector = {3, 1, 8};
11     // Add elements to the end of vector
12     myVector.push_back(5);
13     myVector.push_back(2);
14     // Iterate through all the elements
15     for(int i = 0; i < 5; i++)
16         cout << myVector[i] << endl;
17
18     return 0;
19 }
```

# Add Element to a Vector: push\_back()

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
3  
1  
8  
5  
2
```

```
1  #include <iostream>  
2  // Library required to use vectors  
3  #include <vector>  
4  
5  using namespace std;  
6  
7  int main()  
8  {  
9      // Initialize a vector with some values  
10     vector<int> myVector = {3, 1, 8};  
11     // Add elements to the end of vector  
12     myVector.push_back(5);  
13     myVector.push_back(2);  
14     // Iterate through all the elements  
15     for(int i = 0; i < 5; i++)  
16         cout << myVector[i] << endl;  
17  
18     return 0;  
19 }
```

# Add Element to a Vector: at()

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
??
```

```
1  #include <iostream>
2  // Library required to use vectors
3  #include <vector>
4
5  using namespace std;
6
7  int main()
8  {
9      // Initialize a vector with some values
10     vector<int> myVector = {3, 1, 8};
11     // Add elements to the end of vector
12     myVector.push_back(5);
13     myVector.push_back(2);
14     // Iterate through all the elements
15     for(int i = 0; i < 5; i++)
16         cout << myVector.at(i) << endl;
17
18     return 0;
19 }
```

# Add Element to a Vector: at()

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
3  
1  
8  
5  
2
```

```
1  #include <iostream>  
2  // Library required to use vectors  
3  #include <vector>  
4  
5  using namespace std;  
6  
7  int main()  
8  {  
9      // Initialize a vector with some values  
10     vector<int> myVector = {3, 1, 8};  
11     // Add elements to the end of vector  
12     myVector.push_back(5);  
13     myVector.push_back(2);  
14     // Iterate through all the elements  
15     for(int i = 0; i < 5; i++)  
16         cout << myVector.at(i) << endl;  
17  
18     return 0;  
19 }
```

# Add Element to a Vector: at()

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
3  
1  
8  
5  
2  
exception of type std::out_of_range: vector
```

```
1  #include <iostream>  
2  // Library required to use vectors  
3  #include <vector>  
4  
5  using namespace std;  
6  
7  int main()  
8  {  
9      // Initialize a vector with some values  
10     vector<int> myVector = {3, 1, 8};  
11     // Add elements to the end of vector  
12     myVector.push_back(5);  
13     myVector.push_back(2);  
14     // Iterate through all the elements  
15     for(int i = 0; i < 6; i++)  
16         cout << myVector.at(i) << endl;  
17  
18     return 0;  
19 }
```



# Add Element to a Vector: at()

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
3  
1  
8  
5  
2  
0 (Some garbage value)
```

```
1  #include <iostream>  
2  // Library required to use vectors  
3  #include <vector>  
4  
5  using namespace std;  
6  
7  int main()  
8  {  
9      // Initialize a vector with some values  
10     vector<int> myVector = {3, 1, 8};  
11     // Add elements to the end of vector  
12     myVector.push_back(5);  
13     myVector.push_back(2);  
14     // Iterate through all the elements  
15     for(int i = 0; i < 6; i++)  
16         cout << myVector[i] << endl;  
17  
18     return 0;  
19 }
```



# Delete an Element to a Vector: pop\_back()

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
3  
1  
8  
5
```

```
1  #include <iostream>  
2  // Library required to use vectors  
3  #include <vector>  
4  
5  using namespace std;  
6  
7  int main()  
8  {  
9      // Initialize a vector with some values  
10     vector<int> myVector = {3, 1, 8};  
11     // Add elements to the end of vector  
12     myVector.push_back(5);  
13     myVector.push_back(2);  
14     // Delete the last element of vector  
15     myVector.pop_back();  
16     // Iterate through all the elements  
17     for(int i = 0; i < 4; i++)  
18         cout << myVector.at(i) << endl;  
19  
20     return 0;  
21 }
```

# Get the Number of Elements Present: `size()`

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
??
```

```
5  int main()
6  {
7      // Initialize a vector with some values
8      vector<int> myVector = {3, 1, 8};
9
10     // Add and delete some elements
11     myVector.push_back(5);
12     myVector.push_back(2);
13     myVector.pop_back();
14     myVector.push_back(10);
15     myVector.pop_back();
16     myVector.pop_back();
17     myVector.pop_back();
18
19     // Get the number of elements present
20     cout << myVector.size() << endl;
21     return 0;
22 }
```

# Get the Number of Elements Present: `size()`

simple\_vector.cpp

```
$ g++ -std=c++14 program simple_vector.cpp
```

```
$ ./program
```

```
2
```

```
5  int main()
6  {
7      // Initialize a vector with some values
8      vector<int> myVector = {3, 1, 8};
9
10     // Add and delete some elements
11     myVector.push_back(5);
12     myVector.push_back(2);
13     myVector.pop_back();
14     myVector.push_back(10);
15     myVector.pop_back();
16     myVector.pop_back();
17     myVector.pop_back();
18
19     // Get the number of elements present
20     cout << myVector.size() << endl;
21     return 0;
22 }
```

# Take User Inputs to Put it in a Vector

user\_inputs.cpp

```
$ g++ -std=c++14 program user_inputs.cpp
```

```
$ ./program
```

```
5  
12  
25  
0  
3  
-7
```

```
5 int main()  
6 {  
7     // A vector of unsigned ints (0 and positives)  
8     vector<unsigned int> myVector;  
9     // An infinite loop to take user input  
10    // until the user inserts a negative value  
11    while(true)  
12    {  
13        int temp;  
14        cin >> temp;  
15        if(temp >= 0)  
16            myVector.push_back(temp);  
17        else  
18            break;  
19    }  
20    // Get the size of the vector  
21    int vectorSize = myVector.size();  
22    for(int i = 0; i < vectorSize; i++)  
23        cout << myVector[i] << endl;  
24  
25    return 0;  
26 }
```

# Take User Inputs to Put it in a Vector

user\_inputs.cpp

```
$ g++ -std=c++14 program user_inputs.cpp
```

```
$ ./program
```

```
5
12
25
0
3
-7
5
12
25
0
3
```

```
5 int main()
6 {
7     // A vector of unsigned ints (0 and positives)
8     vector<unsigned int> myVector;
9     // An infinite loop to take user input
10    // until the user inserts a negative value
11    while(true)
12    {
13        int temp;
14        cin >> temp;
15        if(temp >= 0)
16            myVector.push_back(temp);
17        else
18            break;
19    }
20    // Get the size of the vector
21    int vectorSize = myVector.size();
22    for(int i = 0; i < vectorSize; i++)
23        cout << myVector[i] << endl;
24
25    return 0;
26 }
```

# (Some of the) C++ Vector Functions

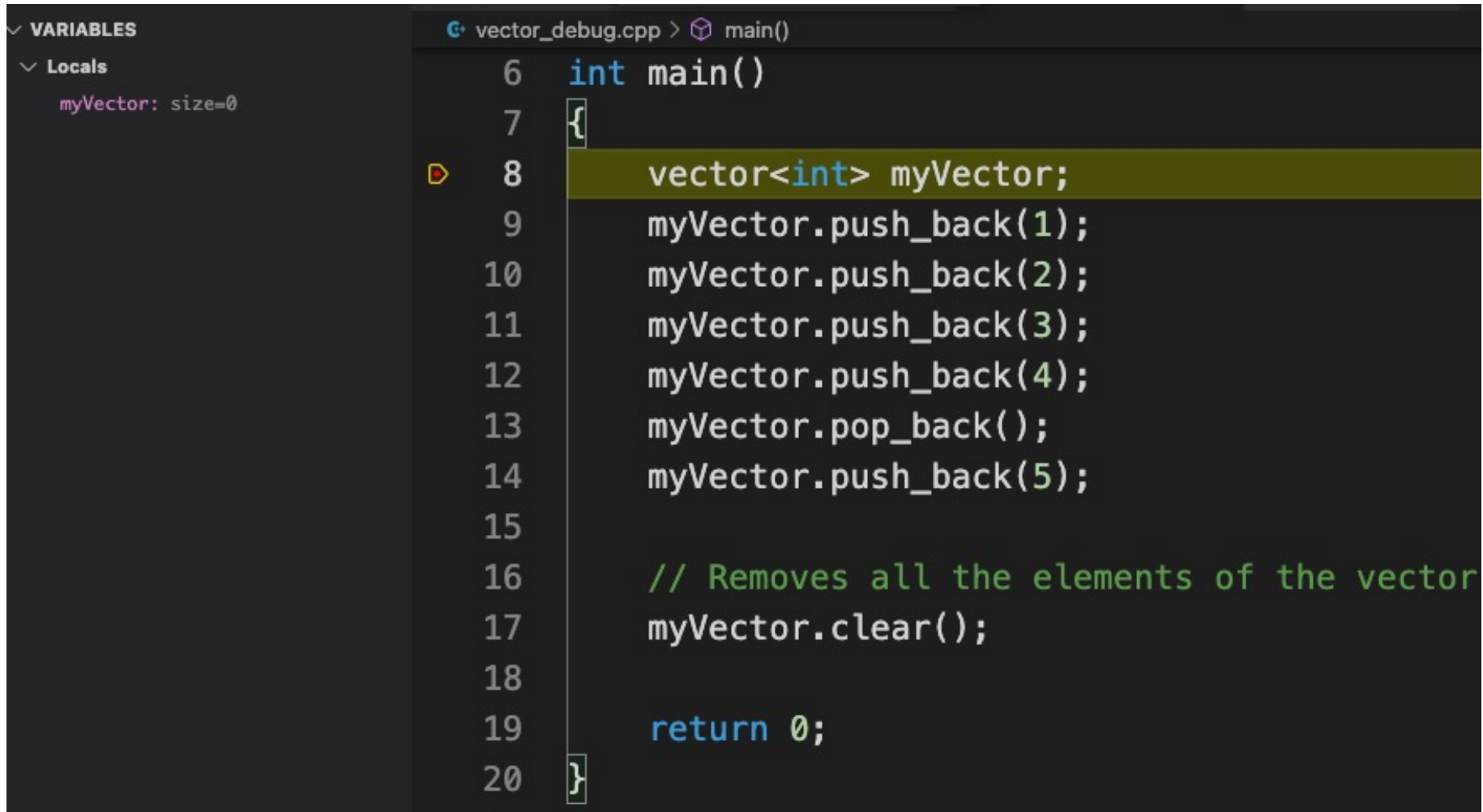
Function	Description
<code>size()</code>	returns the number of elements present in the vector
<code>clear()</code>	removes all the elements of the vector
<code>front()</code>	returns the first element of the vector
<code>back()</code>	returns the last element of the vector
<code>empty()</code>	returns <b>1</b> (true) if the vector is empty
<code>capacity()</code>	check the overall size of a vector

# Vector: Debug Mode

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  int main()
7  {
8      vector<int> myVector;
9      myVector.push_back(1);
10     myVector.push_back(2);
11     myVector.push_back(3);
12     myVector.push_back(4);
13     myVector.pop_back();
14     myVector.push_back(5);
15
16     // Removes all the elements of the vector
17     myVector.clear();
18
19     return 0;
20 }
```



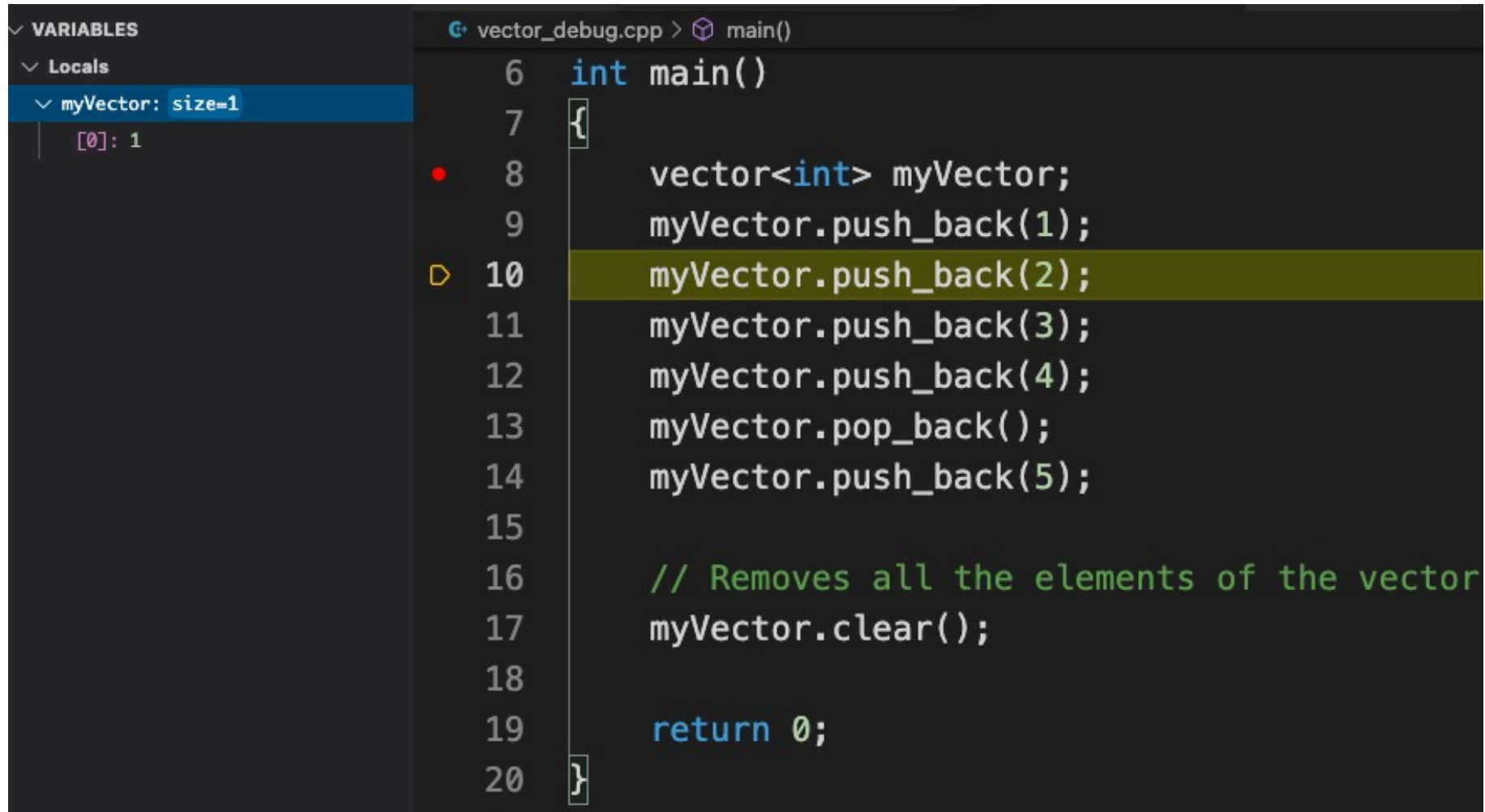
# Vector: Debug Mode



```
vector_debug.cpp > main()
6  int main()
7  {
8  vector<int> myVector;
9  myVector.push_back(1);
10 myVector.push_back(2);
11 myVector.push_back(3);
12 myVector.push_back(4);
13 myVector.pop_back();
14 myVector.push_back(5);
15
16 // Removes all the elements of the vector
17 myVector.clear();
18
19 return 0;
20 }
```



# Vector: Debug Mode

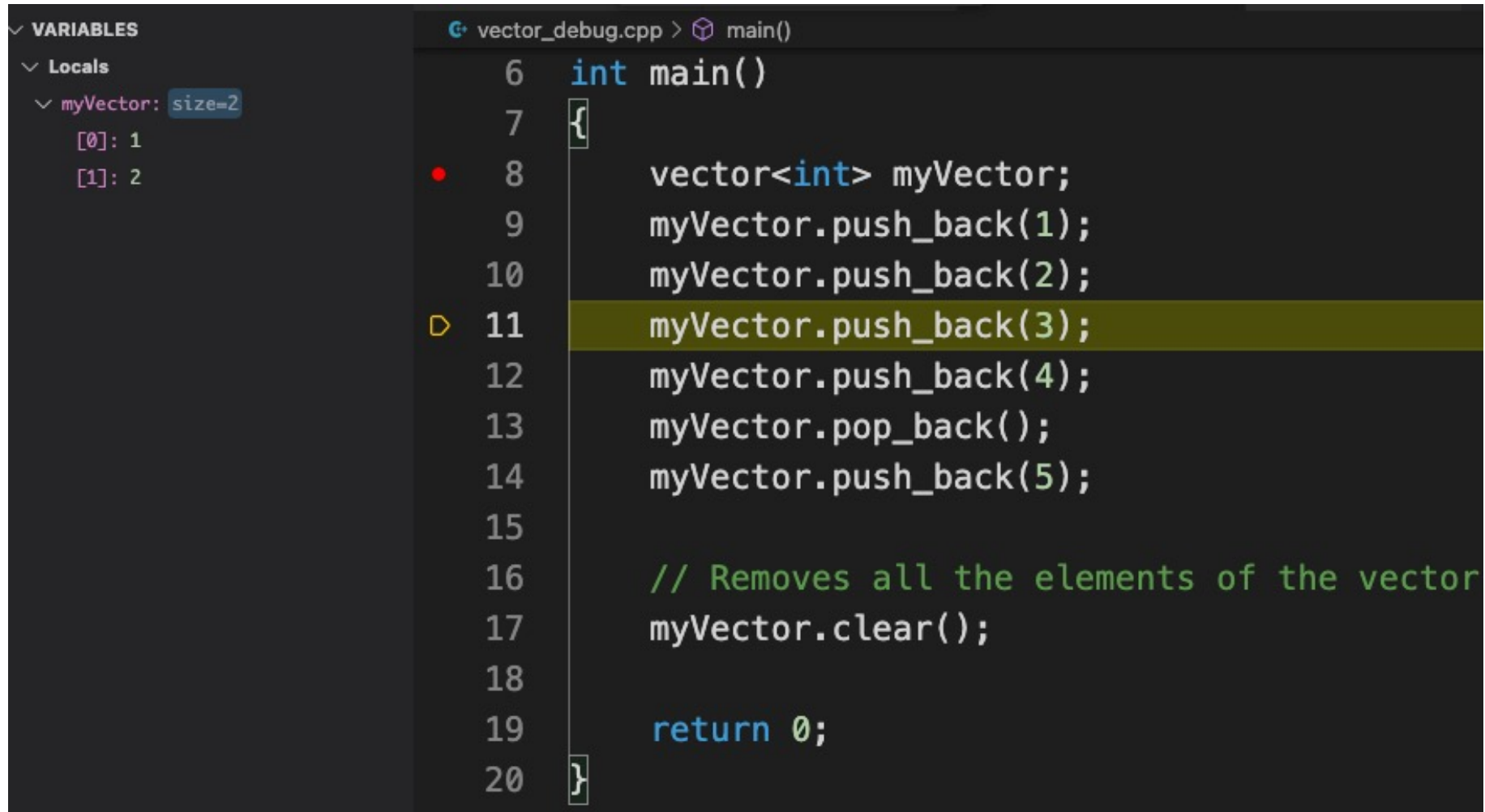


```
vector_debug.cpp > main()
6  int main()
7  {
8      vector<int> myVector;
9      myVector.push_back(1);
10     myVector.push_back(2);
11     myVector.push_back(3);
12     myVector.push_back(4);
13     myVector.pop_back();
14     myVector.push_back(5);
15
16     // Removes all the elements of the vector
17     myVector.clear();
18
19     return 0;
20 }
```

**VARIABLES**

- Locals
  - myVector: size=1
    - [0]: 1

# Vector: Debug Mode



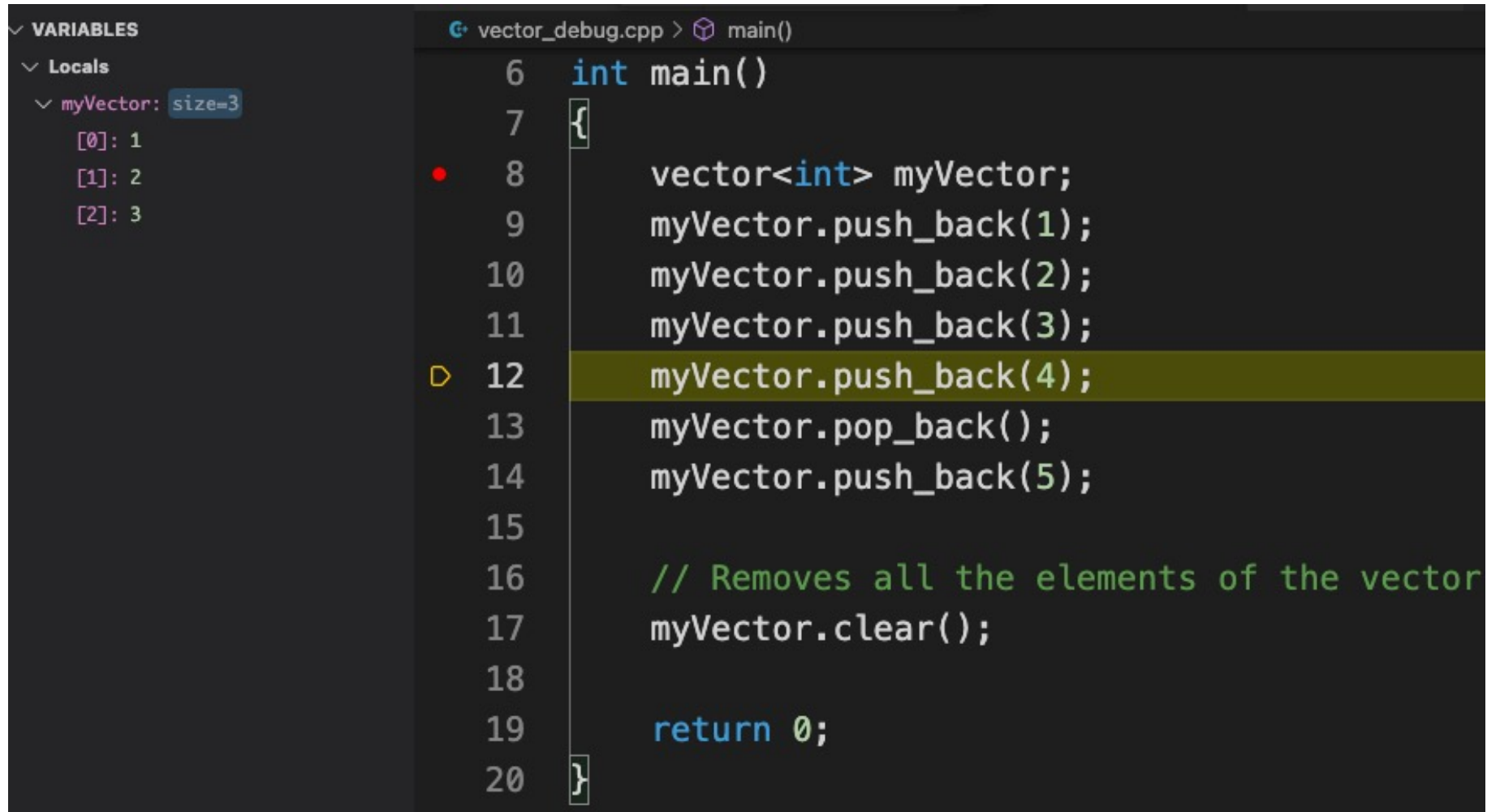
```
vector_debug.cpp > main()

6  int main()
7  {
8      vector<int> myVector;
9      myVector.push_back(1);
10     myVector.push_back(2);
11     myVector.push_back(3);
12     myVector.push_back(4);
13     myVector.pop_back();
14     myVector.push_back(5);
15
16     // Removes all the elements of the vector
17     myVector.clear();
18
19     return 0;
20 }
```

**VARIABLES**

- Locals
  - myVector: size=2
    - [0]: 1
    - [1]: 2

# Vector: Debug Mode

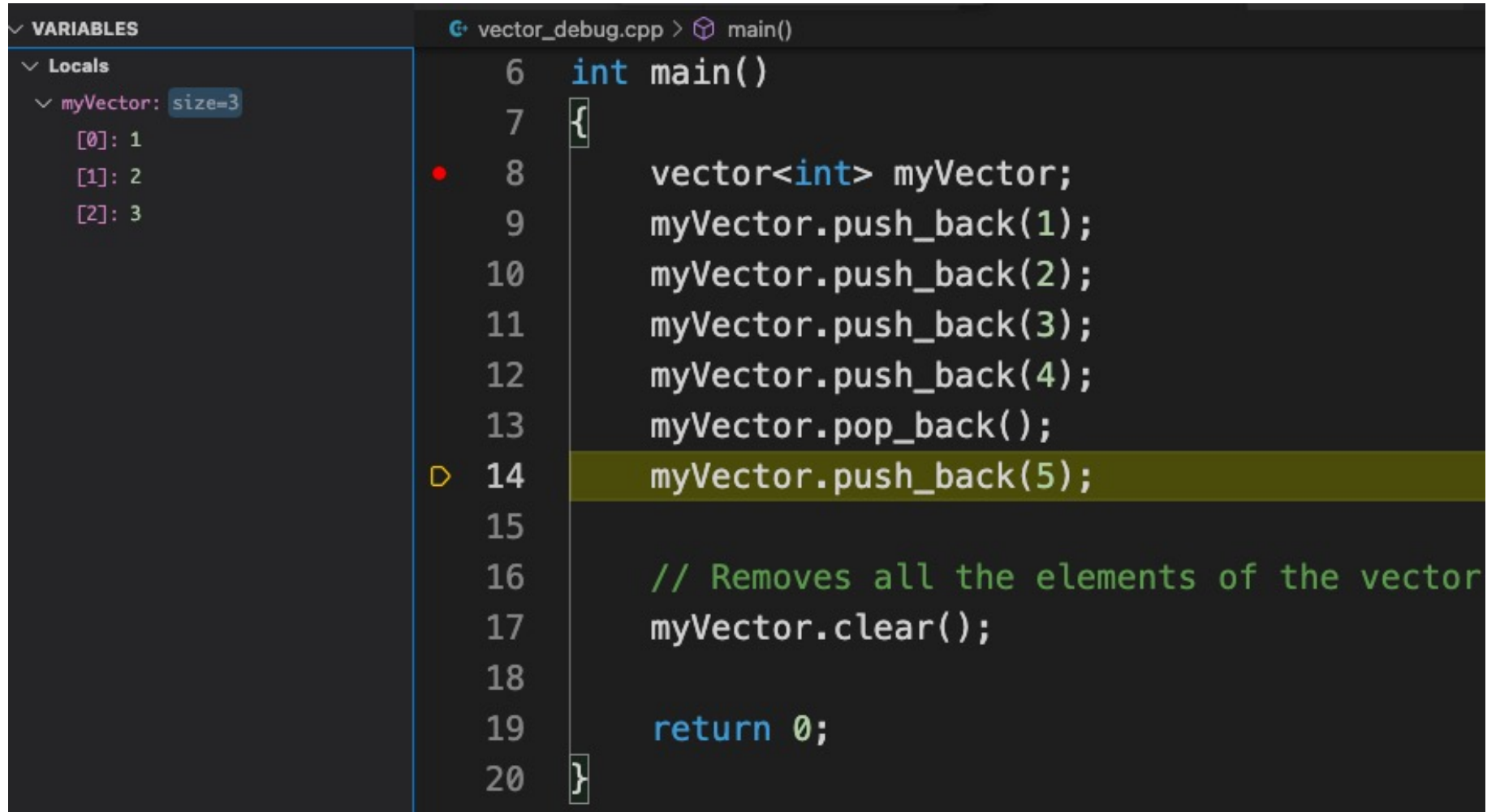


```
vector_debug.cpp > main()
6  int main()
7  {
8      vector<int> myVector;
9      myVector.push_back(1);
10     myVector.push_back(2);
11     myVector.push_back(3);
12     myVector.push_back(4);
13     myVector.pop_back();
14     myVector.push_back(5);
15
16     // Removes all the elements of the vector
17     myVector.clear();
18
19     return 0;
20 }
```

# Vector: Debug Mode

```
vector_debug.cpp > main()
6  int main()
7  {
8      vector<int> myVector;
9      myVector.push_back(1);
10     myVector.push_back(2);
11     myVector.push_back(3);
12     myVector.push_back(4);
13     myVector.pop_back();
14     myVector.push_back(5);
15
16     // Removes all the elements of the vector
17     myVector.clear();
18
19     return 0;
20 }
```

# Vector: Debug Mode



```
vector_debug.cpp > main()
6  int main()
7  {
8      vector<int> myVector;
9      myVector.push_back(1);
10     myVector.push_back(2);
11     myVector.push_back(3);
12     myVector.push_back(4);
13     myVector.pop_back();
14     myVector.push_back(5);
15
16     // Removes all the elements of the vector
17     myVector.clear();
18
19     return 0;
20 }
```

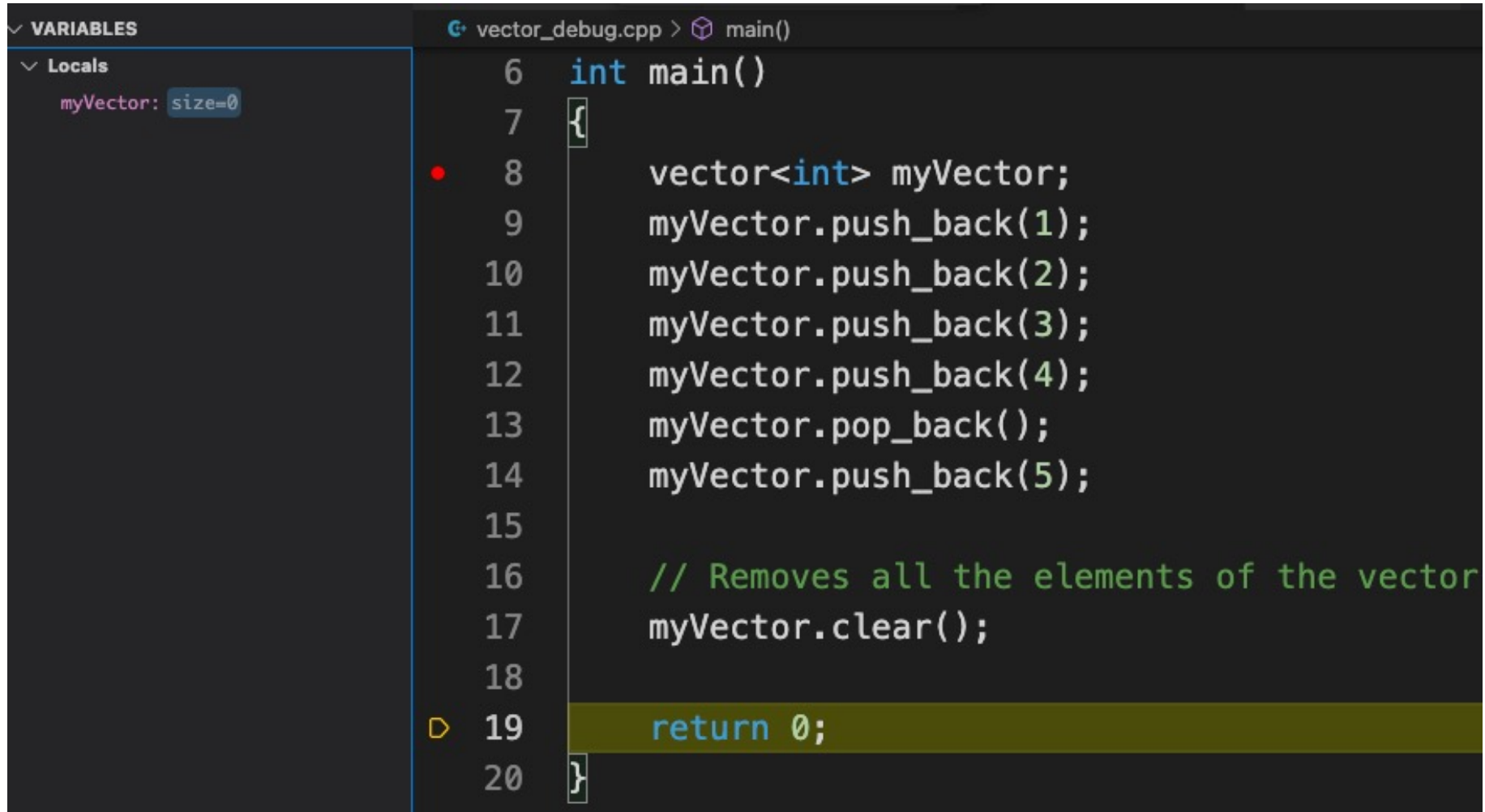
**VARIABLES**

- Locals
  - myVector: size=3
    - [0]: 1
    - [1]: 2
    - [2]: 3

# Vector: Debug Mode

```
vector_debug.cpp > main()
6  int main()
7  {
8      vector<int> myVector;
9      myVector.push_back(1);
10     myVector.push_back(2);
11     myVector.push_back(3);
12     myVector.push_back(4);
13     myVector.pop_back();
14     myVector.push_back(5);
15
16     // Removes all the elements of the vector
17     myVector.clear();
18
19     return 0;
20 }
```

# Vector: Debug Mode



```
vector_debug.cpp > main()
6  int main()
7  {
8      vector<int> myVector;
9      myVector.push_back(1);
10     myVector.push_back(2);
11     myVector.push_back(3);
12     myVector.push_back(4);
13     myVector.pop_back();
14     myVector.push_back(5);
15
16     // Removes all the elements of the vector
17     myVector.clear();
18
19     return 0;
20 }
```

**VARIABLES**

Locals

myVector: size=0



# Vector: Pass by Value to a Function

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  void DisplayVector(vector<int> iVectors)
6  {
7      int vectorSize = iVectors.size();
8      for(int i = 0; i < vectorSize; i++)
9          cout << iVectors.at(i) << endl;
10 }
11
12 int main()
13 {
14     vector<int> myVector;
15     for(int i = 1; i <= 5; i++)
16         myVector.push_back(i*10);
17
18     DisplayVector(myVector);
19     return 0;
20 }
```

O/P ?

??



# Vector: Pass by Value to a Function

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  void DisplayVector(vector<int> iVectors)
6  {
7      int vectorSize = iVectors.size();
8      for(int i = 0; i < vectorSize; i++)
9          cout << iVectors.at(i) << endl;
10 }
11
12 int main()
13 {
14     vector<int> myVector;
15     for(int i = 1; i <= 5; i++)
16         myVector.push_back(i*10);
17
18     DisplayVector(myVector);
19     return 0;
20 }
```

O/P ↴

10
20
30
40
50

# Vector: Pass by Value to a Function

```
5 void foo(vector<int> iVectors)
6 {
7     iVectors.push_back(60);
8 }
9
10 int main()
11 {
12     vector<int> myVector;
13     for(int i = 1; i <= 5; i++)
14         myVector.push_back(i*10);
15
16     foo(myVector);
17
18     int vectorSize = myVector.size();
19     for(int i = 0; i < vectorSize; i++)
20         cout << myVector.at(i) << endl;
21
22     return 0;
23 }
```

O/P 2  
??

# Vector: Pass by Value to a Function

**Note:** Unlike Arrays, vectors are not always passed by reference.

```
5 void foo(vector<int> iVectors)
6 {
7     iVectors.push_back(60);
8 }
9
10 int main()
11 {
12     vector<int> myVector;
13     for(int i = 1; i <= 5; i++)
14         myVector.push_back(i*10);
15
16     foo(myVector);
17
18     int vectorSize = myVector.size();
19     for(int i = 0; i < vectorSize; i++)
20         cout << myVector.at(i) << endl;
21
22     return 0;
23 }
```

O/P

10
20
30
40
50

# Vector: Pass by Value to a Function

```
5  vector<int> foo(vector<int> iVectors)
6  {
7      iVectors.push_back(60);
8      ↗ return iVectors;
9  }
10
11 int main()
12 {
13     vector<int> myVector;
14     for(int i = 1; i <= 5; i++)
15         myVector.push_back(i*10);
16
17     ↗ myVector = foo(myVector);
18
19     int vectorSize = myVector.size();
20     for(int i = 0; i < vectorSize; i++)
21         cout << myVector.at(i) << endl;
22
23     return 0;
24 }
```

o/p B  
??

# Vector: Pass by Value to a Function

```
5  vector<int> foo(vector<int> iVectors)
6  {
7      iVectors.push_back(60);
8      ↗ return iVectors;
9  }
10
11 int main()
12 {
13     vector<int> myVector;
14     for(int i = 1; i <= 5; i++)
15         myVector.push_back(i*10);
16
17     ↗ myVector = foo(myVector);
18
19     int vectorSize = myVector.size();
20     for(int i = 0; i < vectorSize; i++)
21         cout << myVector.at(i) << endl;
22
23     return 0;
24 }
```

o/p ↗

10
20
30
40
50
60

# Vector: Pass by Reference to a Function

```
5 void foo(vector<int> &iVectors)
6 {
7     iVectors.push_back(60);
8 }
9
10 int main()
11 {
12     vector<int> myVector;
13     for(int i = 1; i <= 5; i++)
14         myVector.push_back(i*10);
15
16     foo(myVector);
17
18     int vectorSize = myVector.size();
19     for(int i = 0; i < vectorSize; i++)
20         cout << myVector.at(i) << endl;
21
22     return 0;
23 }
```

O/P 2  
??

# Vector: Pass by Reference to a Function

```
5 void foo(vector<int> &iVectors)
6 {
7     iVectors.push_back(60);
8 }
9
10 int main()
11 {
12     vector<int> myVector;
13     for(int i = 1; i <= 5; i++)
14         myVector.push_back(i*10);
15
16     foo(myVector);
17
18     int vectorSize = myVector.size();
19     for(int i = 0; i < vectorSize; i++)
20         cout << myVector.at(i) << endl;
21
22     return 0;
23 }
```

O/P

10
20
30
40
50
60

# for-each Loop for Vectors

for\_each\_loop.cpp

```
$ g++ -std=c++14 program for_each_loop.cpp
```

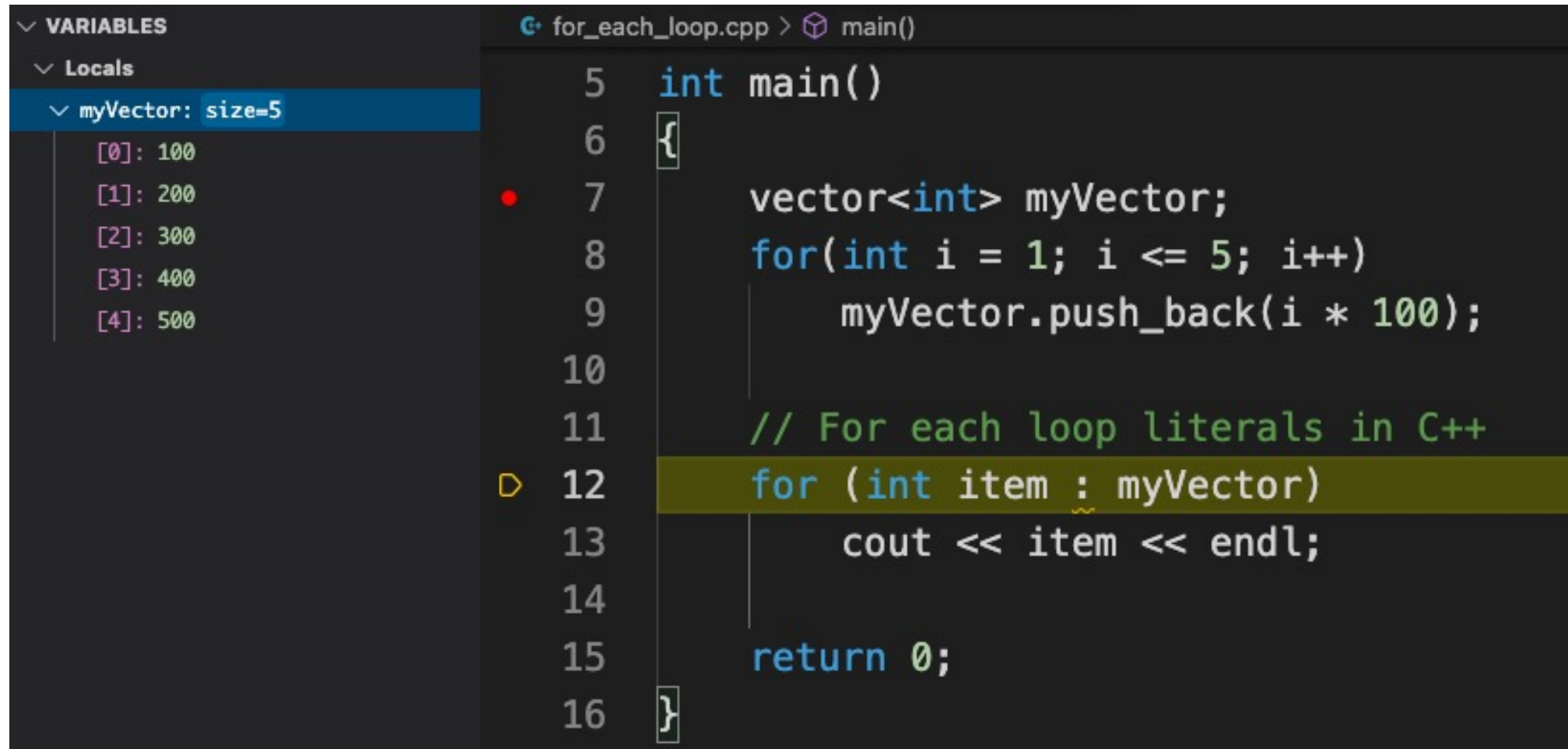
```
$ ./program
```

```
100  
200  
300  
400  
500
```

```
1  #include <iostream>  
2  #include <vector>  
3  using namespace std;  
4  
5  int main()  
6  {  
7      vector<int> myVector;  
8      for(int i = 1; i <= 5; i++)  
9          myVector.push_back(i * 100);  
10  
11     // For each loop literals in C++  
12     for (int item : myVector)  
13         cout << item << endl;  
14  
15     return 0;  
16 }
```



# for-each Loop for Vectors: Debug



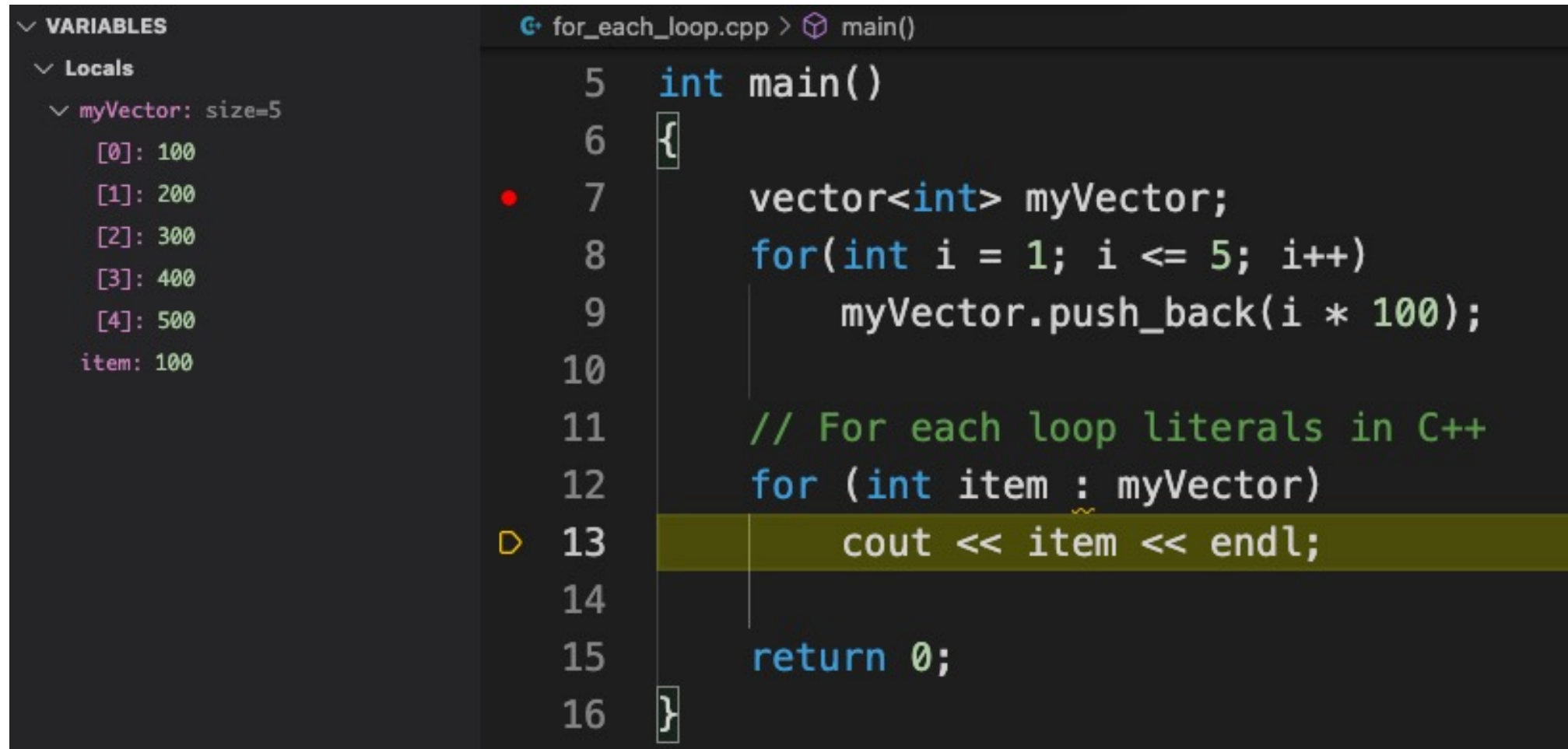
The screenshot shows a C++ IDE with a file named `for_each_loop.cpp` and a function `main()`. The code defines a `vector<int>` named `myVector` and populates it with values from 100 to 500 in increments of 100. A `for` loop with range-comprehension syntax (`for (int item : myVector)`) is used to iterate over the vector and print each element. The IDE's `VARIABLES` pane on the left shows the state of `myVector` with its size and the values of its elements at indices 0 through 4. The code is as follows:

```
5 int main()
6 {
7     vector<int> myVector;
8     for(int i = 1; i <= 5; i++)
9         myVector.push_back(i * 100);
10
11     // For each loop literals in C++
12     for (int item : myVector)
13         cout << item << endl;
14
15     return 0;
16 }
```

The `VARIABLES` pane shows:

- `myVector: size=5`
- `[0]: 100`
- `[1]: 200`
- `[2]: 300`
- `[3]: 400`
- `[4]: 500`

# for-each Loop for Vectors: Debug



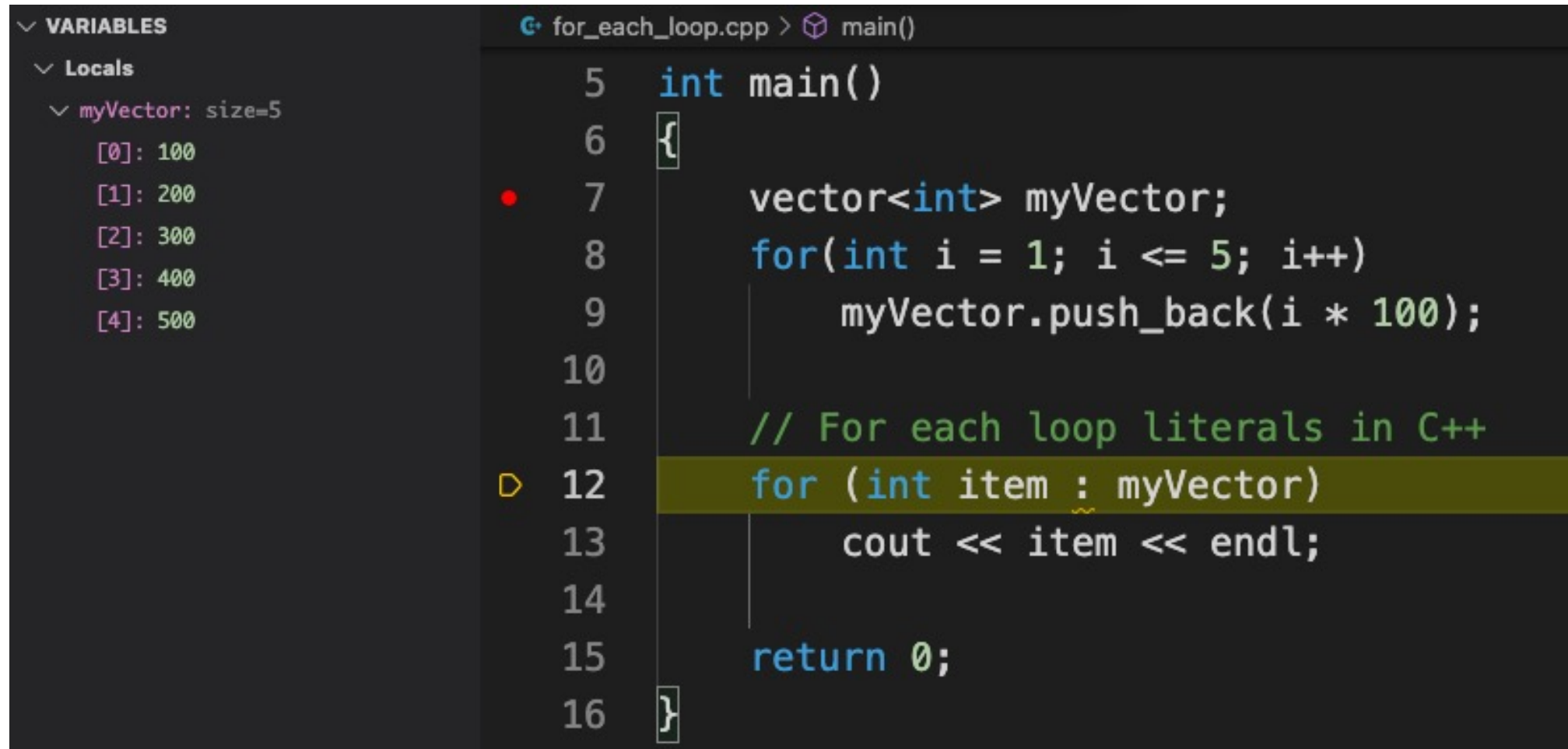
The screenshot shows a C++ IDE with a file named `for_each_loop.cpp` in the `main()` function. The code defines a `vector<int>` named `myVector` and populates it with values from 100 to 500 in a loop. A `for` loop with a range-based `for` loop (for-each) is used to iterate over `myVector`, printing each element. The line `cout << item << endl;` is highlighted. On the left, the `VARIABLES` pane shows the state of `myVector` and the current `item`.

```
5  int main()
6  {
7      vector<int> myVector;
8      for(int i = 1; i <= 5; i++)
9          myVector.push_back(i * 100);
10
11     // For each loop literals in C++
12     for (int item : myVector)
13         cout << item << endl;
14
15     return 0;
16 }
```

**VARIABLES**

- Locals
  - myVector: size=5
    - [0]: 100
    - [1]: 200
    - [2]: 300
    - [3]: 400
    - [4]: 500
  - item: 100

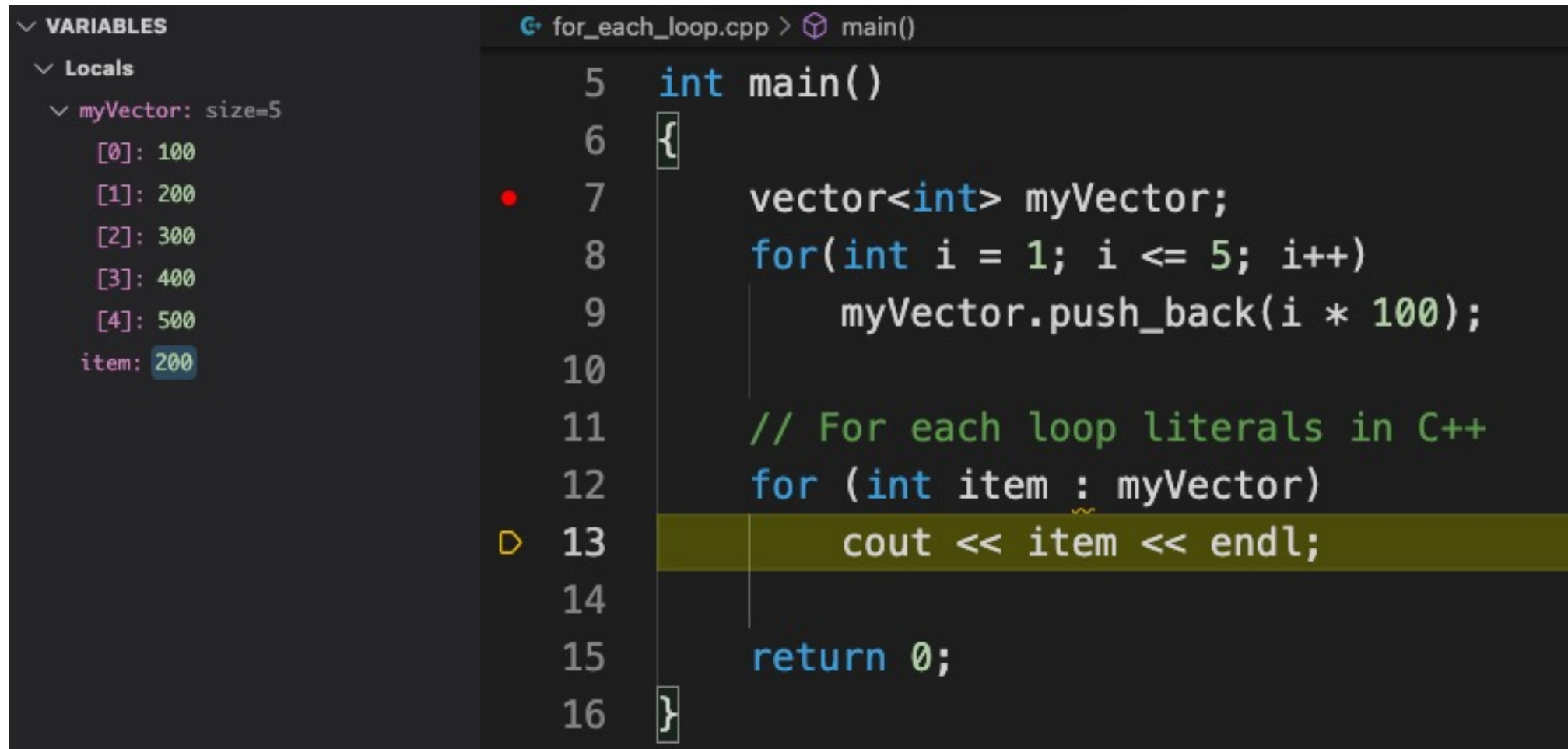
# for-each Loop for Vectors: Debug



The screenshot shows a C++ IDE with a dark theme. On the left, the 'VARIABLES' pane is expanded to 'Locals', showing a variable 'myVector' of type 'size=5'. Below it, the elements of the vector are listed: [0]: 100, [1]: 200, [2]: 300, [3]: 400, and [4]: 500. The main editor shows the source code for 'for\_each\_loop.cpp' in the 'main()' function. The code defines a 'vector<int> myVector;', pushes back values from 100 to 500 in a loop, and then uses a 'for (int item : myVector)' loop to print each element. The 'for' loop line is highlighted in yellow. A red dot on line 7 indicates a breakpoint, and a yellow arrow on line 12 indicates the current execution point.

```
5  int main()
6  {
7      vector<int> myVector;
8      for(int i = 1; i <= 5; i++)
9          myVector.push_back(i * 100);
10
11     // For each loop literals in C++
12     for (int item : myVector)
13         cout << item << endl;
14
15     return 0;
16 }
```

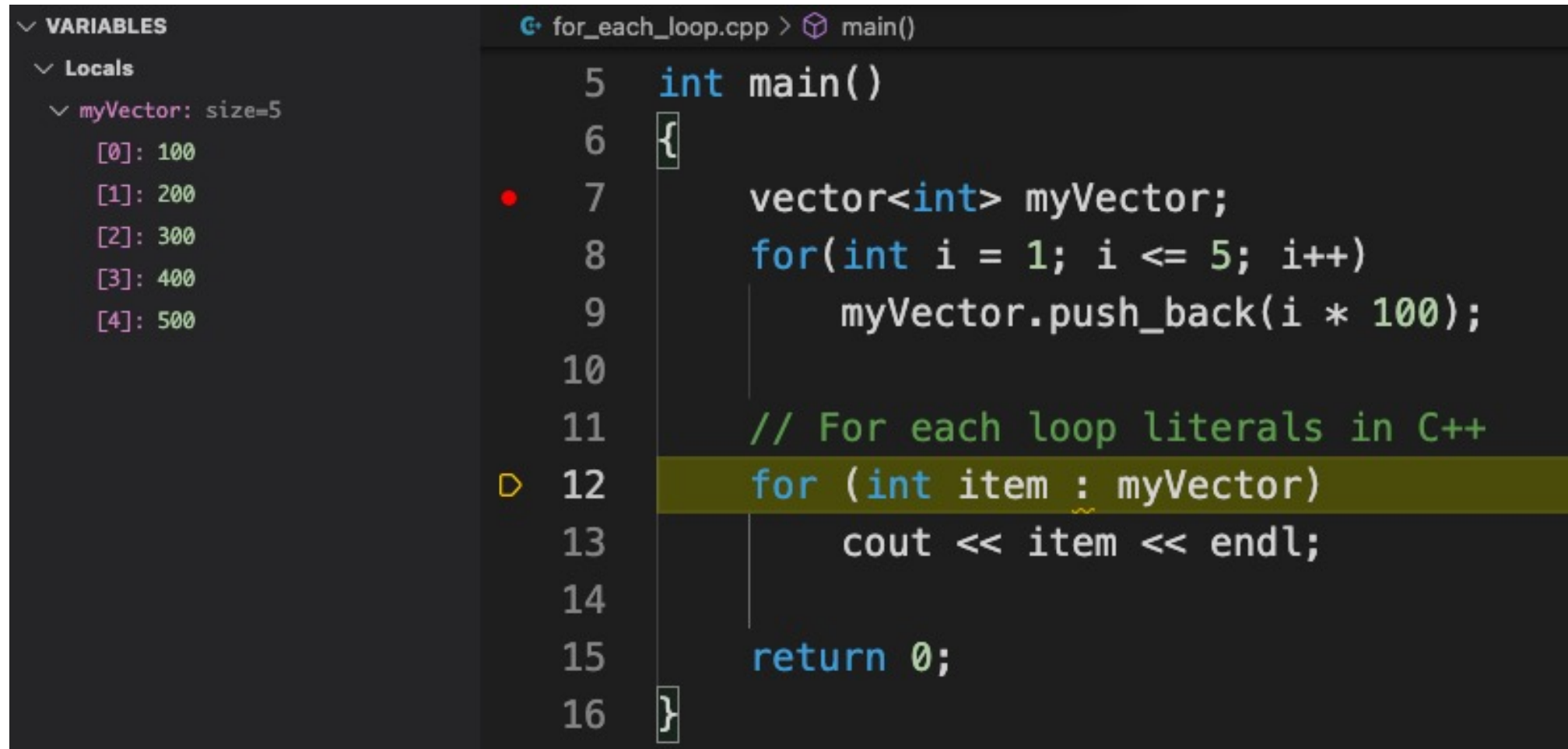
# for-each Loop for Vectors: Debug



The screenshot shows a C++ IDE with a dark theme. On the left, the 'VARIABLES' pane is expanded to 'Locals', showing a vector named 'myVector' with a size of 5. The elements are listed as [0]: 100, [1]: 200, [2]: 300, [3]: 400, and [4]: 500. Below this, a variable 'item' is shown with a value of 200. The main editor displays the source code for 'for\_each\_loop.cpp' in the 'main()' function. The code includes a vector declaration, a loop to push back values from 1 to 5 multiplied by 100, a comment about for-each loop literals, and a for-each loop that iterates over 'myVector' and prints each 'item'. The line 'cout << item << endl;' is highlighted in yellow, and a yellow arrow points to it from the left margin. A red dot is visible on line 7, and a yellow arrow points to it from the left margin.

```
5 int main()
6 {
7     vector<int> myVector;
8     for(int i = 1; i <= 5; i++)
9         myVector.push_back(i * 100);
10
11     // For each loop literals in C++
12     for (int item : myVector)
13         cout << item << endl;
14
15     return 0;
16 }
```

# for-each Loop for Vectors: Debug



The screenshot shows a C++ IDE with a dark theme. On the left, the 'VARIABLES' pane is expanded to 'Locals', showing a variable 'myVector' of type 'size=5'. Below it, the elements of the vector are listed: [0]: 100, [1]: 200, [2]: 300, [3]: 400, and [4]: 500. The main editor shows the code for 'for\_each\_loop.cpp' in the 'main()' function. The code is as follows:

```
5 int main()
6 {
7     vector<int> myVector;
8     for(int i = 1; i <= 5; i++)
9         myVector.push_back(i * 100);
10
11     // For each loop literals in C++
12     for (int item : myVector)
13         cout << item << endl;
14
15     return 0;
16 }
```

The line containing the for-each loop (line 12) is highlighted in yellow. A yellow arrow points to the start of the loop, and a red dot is on line 7. The IDE title bar shows 'for\_each\_loop.cpp' and 'main()'.

# for-each Loop for Vectors: Debug

```
for_each_loop.cpp > main()

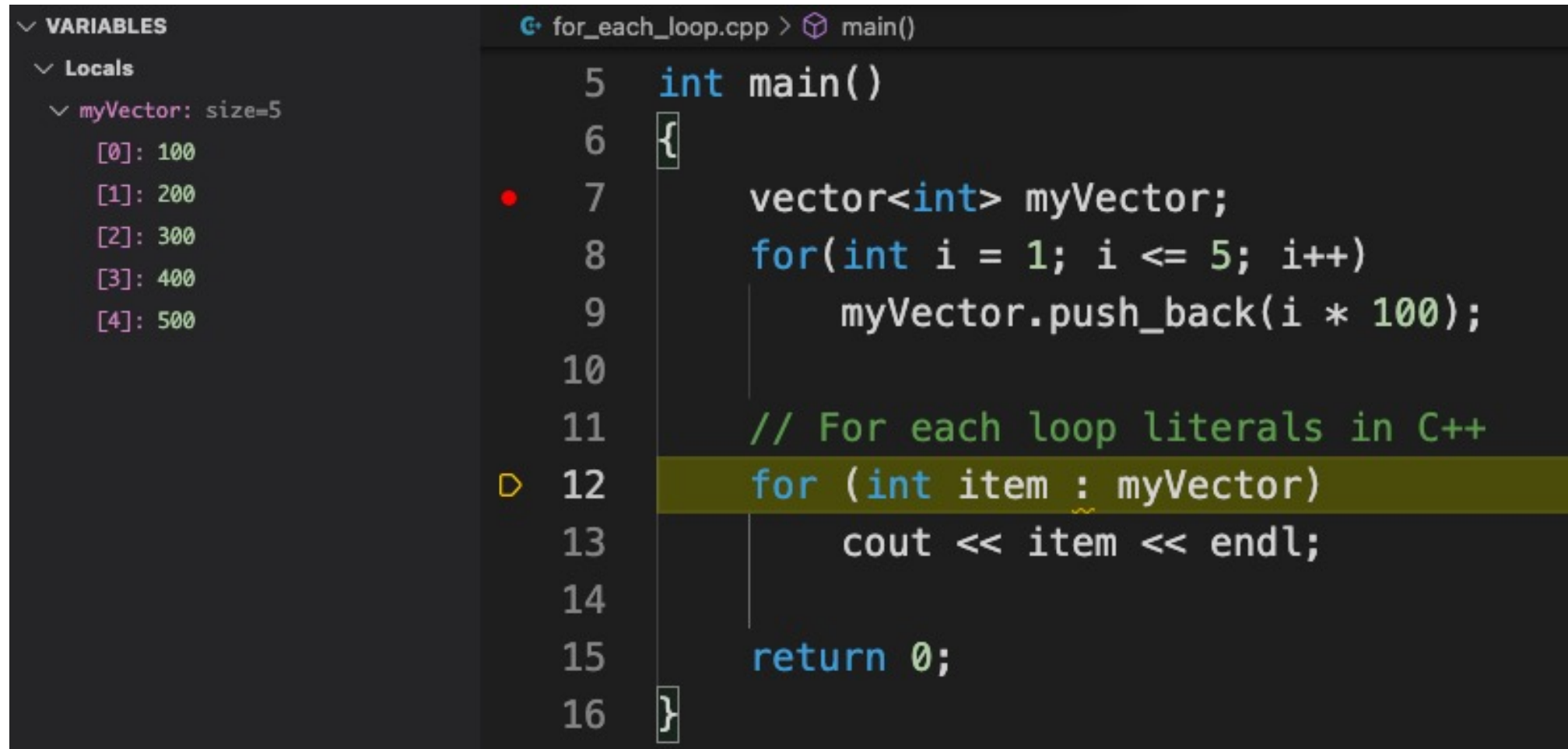
5  int main()
6  {
7      vector<int> myVector;
8      for(int i = 1; i <= 5; i++)
9          myVector.push_back(i * 100);
10
11      // For each loop literals in C++
12      for (int item : myVector)
13          cout << item << endl;
14
15      return 0;
16 }
```

**VARIABLES**

- Locals
  - myVector: size=5
    - [0]: 100
    - [1]: 200
    - [2]: 300
    - [3]: 400
    - [4]: 500
  - item: 300



# for-each Loop for Vectors: Debug



The screenshot shows a C++ IDE with a dark theme. On the left, the 'VARIABLES' pane is expanded to 'Locals', showing a variable 'myVector' of type 'size=5'. Below it, the elements of the vector are listed: [0]: 100, [1]: 200, [2]: 300, [3]: 400, and [4]: 500. The main editor shows the code for 'for\_each\_loop.cpp' in the 'main()' function. The code is as follows:

```
5 int main()
6 {
7     vector<int> myVector;
8     for(int i = 1; i <= 5; i++)
9         myVector.push_back(i * 100);
10
11     // For each loop literals in C++
12     for (int item : myVector)
13         cout << item << endl;
14
15     return 0;
16 }
```

Line 12, which contains the for-each loop, is highlighted in green. A yellow arrow points to the 'for' keyword, and a red dot is on line 7. The comment on line 11 reads '// For each loop literals in C++'.

# for-each Loop for Vectors: Debug

```

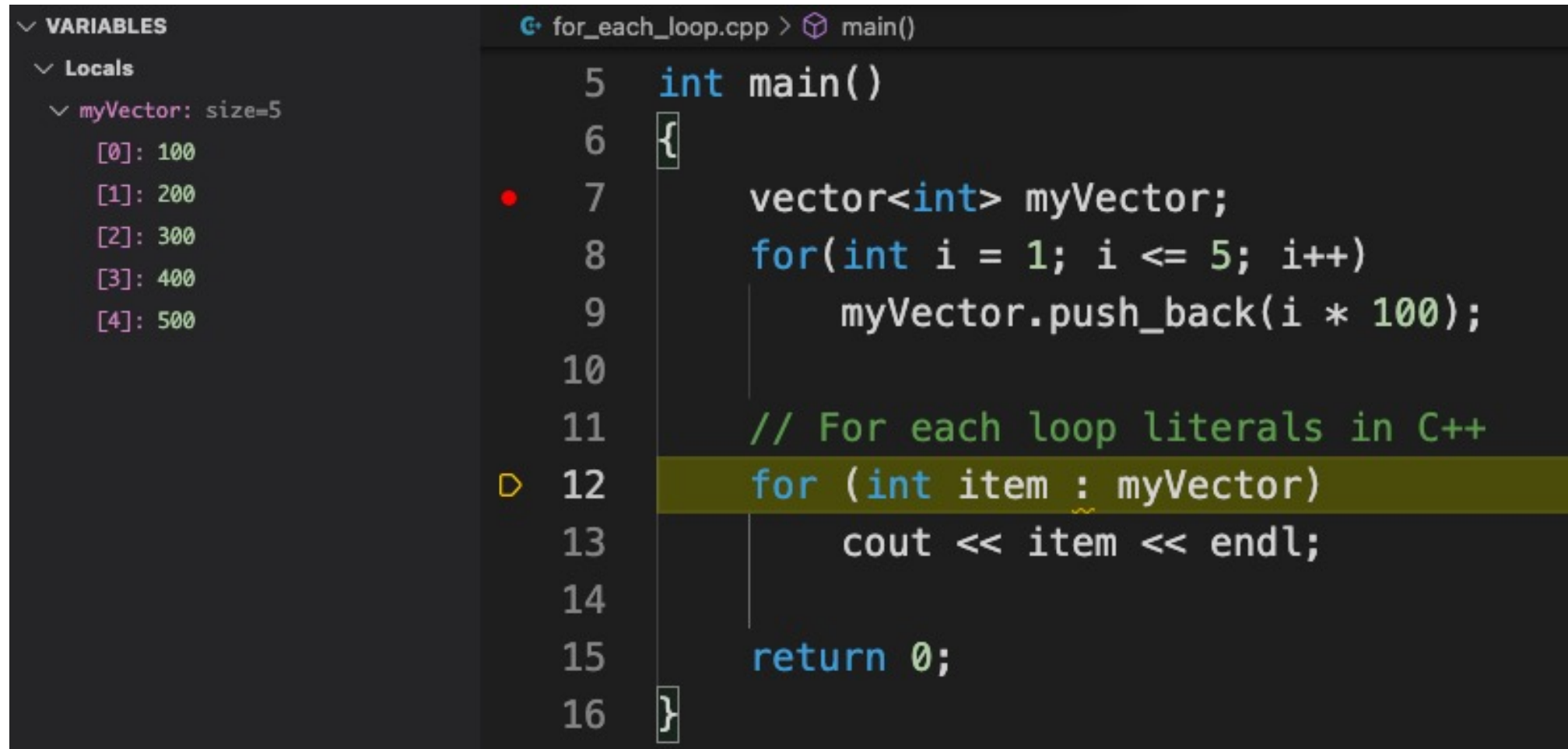
5  int main()
6  {
7      vector<int> myVector;
8      for(int i = 1; i <= 5; i++)
9          myVector.push_back(i * 100);
10
11      // For each loop literals in C++
12      for (int item : myVector)
13          cout << item << endl;
14
15      return 0;
16  }
```

**VARIABLES**

- Locals
  - myVector: size=5
    - [0]: 100
    - [1]: 200
    - [2]: 300
    - [3]: 400
    - [4]: 500
  - item: 400



# for-each Loop for Vectors: Debug



The screenshot shows a C++ IDE with a dark theme. On the left, the 'VARIABLES' pane is expanded to 'Locals', showing a variable 'myVector' of type 'size=5'. Its elements are listed as [0]: 100, [1]: 200, [2]: 300, [3]: 400, and [4]: 500. The main editor displays the code for 'for\_each\_loop.cpp' in the 'main()' function. The code includes a vector declaration, a loop to push back values from 100 to 500, a comment about C++ range literals, and a range-based for loop. The line 'for (int item : myVector)' is highlighted in green, and a yellow arrow points to it, indicating the current execution point. The code ends with 'return 0;'.

```
5 int main()
6 {
7     vector<int> myVector;
8     for(int i = 1; i <= 5; i++)
9         myVector.push_back(i * 100);
10
11     // For each loop literals in C++
12     for (int item : myVector)
13         cout << item << endl;
14
15     return 0;
16 }
```

# for-each Loop for Vectors: Debug

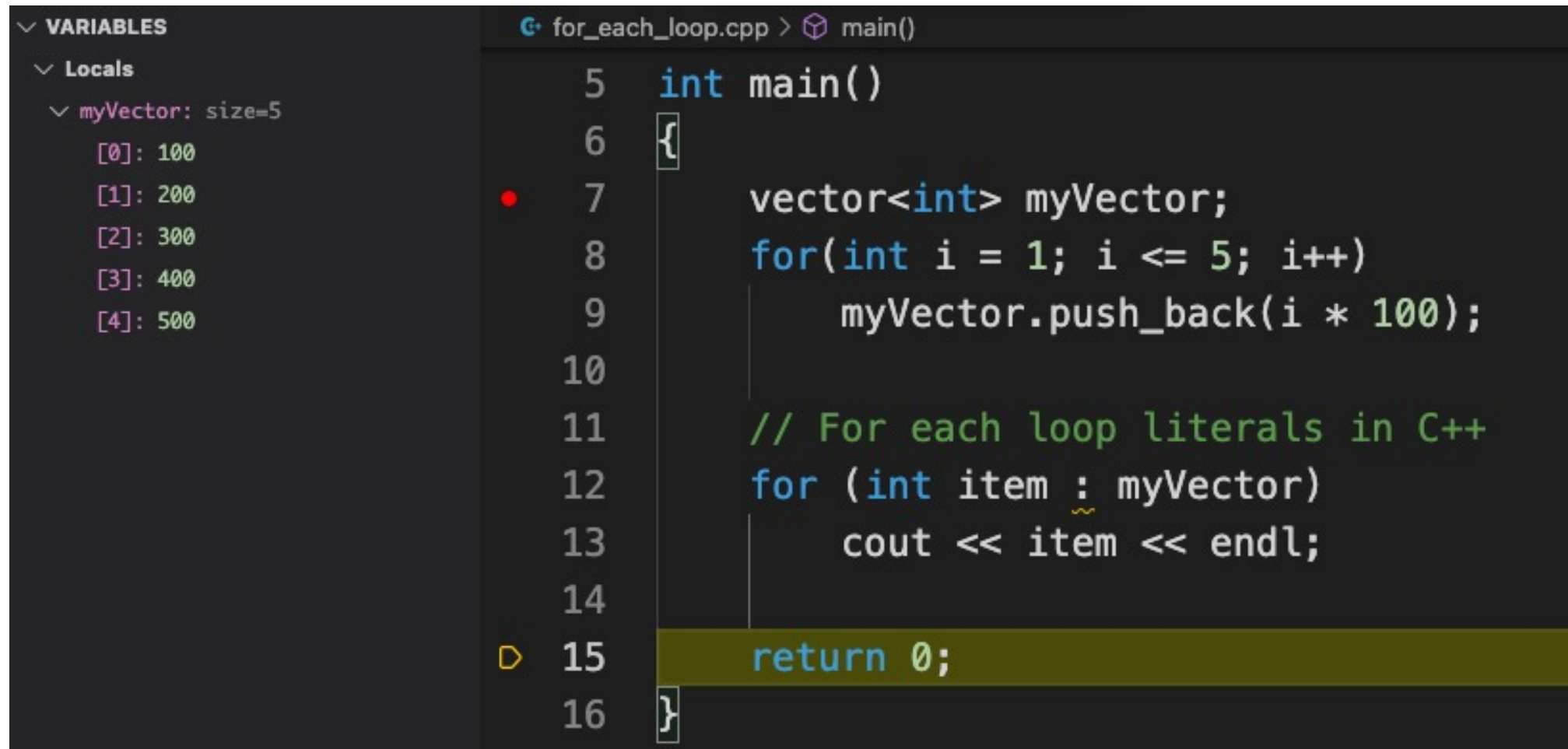
```

5  int main()
6  {
7      vector<int> myVector;
8      for(int i = 1; i <= 5; i++)
9          myVector.push_back(i * 100);
10
11     // For each loop literals in C++
12     for (int item : myVector)
13         cout << item << endl;
14
15     return 0;
16 }
```

**VARIABLES**

- Locals
  - myVector: size=5
    - [0]: 100
    - [1]: 200
    - [2]: 300
    - [3]: 400
    - [4]: 500
  - item: 500

# for-each Loop for Vectors: Debug



The screenshot displays a C++ development environment with two panels. The left panel, titled 'VARIABLES', shows the state of local variables. Under 'Locals', 'myVector' is listed with 'size=5'. Below it, the elements of the vector are shown as an array: [0]: 100, [1]: 200, [2]: 300, [3]: 400, and [4]: 500. The right panel shows the source code for 'for\_each\_loop.cpp' in the 'main()' function. The code defines a vector, fills it with values from 100 to 500 in a loop, and then uses a range-based for loop to iterate over the vector elements and print them. A red dot on line 7 indicates the current execution point. Line 15, 'return 0;', is highlighted with a yellow bar. The code is as follows:

```
5  int main()
6  {
7      vector<int> myVector;
8      for(int i = 1; i <= 5; i++)
9          myVector.push_back(i * 100);
10
11     // For each loop literals in C++
12     for (int item : myVector)
13         cout << item << endl;
14
15     return 0;
16 }
```

# Remarks

- Reference
  - Vectors in C++. Programiz. <https://www.programiz.com/cpp-programming/vectors>
  - CPlusPlus. <https://www.cplusplus.com/reference/vector/vector/>