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

```
#include <iostream>
// Library required to use vectors
#include <vector>
using namespace std;


int main()
{
    // Initialize a vector
    vector <int> myVector = {3, 1, 8};
    // Iterate through each element
    for(int i = 0; i < 3; i++)
    {
        cout << myVector[i] << endl;
    }

    return 0;
}
```

A Simple Program w/ Vector

simple_vector.cpp

```
#include <iostream>
// Library required to use vectors
#include <vector>
using namespace std;


int main()
{
    // Initialize a vector
    vector <int> myVector = {3, 1, 8};
    // Iterate through each element
    for(int i = 0; i < 3; i++)
    {
        cout << myVector[i] << endl;
    }

    return 0;
}
```

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

```
$ ./program
```

A Simple Program w/ Vector

simple_vector.cpp

```
#include <iostream>
// Library required to use vectors
#include <vector>
using namespace std;


int main()
{
    // Initialize a vector
    vector <int> myVector = {3, 1, 8};
    // Iterate through each element
    for(int i = 0; i < 3; i++)
    {
        cout << myVector[i] << endl;
    }

    return 0;
}
```

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

```
$ ./program
```

```
3
1
8
```

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
```

```
??
```

```cpp
#include <iostream>
// Library required to use vectors
#include <vector>

using namespace std;

int main()
{
    // Initialize a vector with some values
    vector<int> myVector = {3, 1, 8};
    // Add elements to the end of vector
    myVector.push_back(5);
    myVector.push_back(2);
    // Iterate through all the elements
    for(int i = 0; i < 5; i++)
        cout << myVector[i] << endl;

    return 0;
}
```

Add Element to a Vector: push_back()

simple_vector.cpp

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

```
$ ./program
```

```
3
1
8
5
2
```

```cpp
#include <iostream>
// Library required to use vectors
#include <vector>

using namespace std;

int main()
{
    // Initialize a vector with some values
    vector<int> myVector = {3, 1, 8};
    // Add elements to the end of vector
    myVector.push_back(5);
    myVector.push_back(2);
    // Iterate through all the elements
    for(int i = 0; i < 5; i++)
        cout << myVector[i] << endl;

    return 0;
}
```

Add Element to a Vector: at()

simple_vector.cpp

```
#include <iostream>
// Library required to use vectors
#include <vector>

using namespace std;

int main()
{
    // Initialize a vector with some values
    vector<int> myVector = {3, 1, 8};
    // Add elements to the end of vector
    myVector.push_back(5);
    myVector.push_back(2);
    // Iterate through all the elements
    for(int i = 0; i < 5; i++)
        cout << myVector.at(i) << endl;

    return 0;
}
```

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

```
$ ./program
```

```
??
```

Add Element to a Vector: at()

simple_vector.cpp

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

```
$ ./program
```

```
3
1
8
5
2
```

```cpp
#include <iostream>
// Library required to use vectors
#include <vector>

using namespace std;

int main()
{
    // Initialize a vector with some values
    vector<int> myVector = {3, 1, 8};
    // Add elements to the end of vector
    myVector.push_back(5);
    myVector.push_back(2);
    // Iterate through all the elements
    for(int i = 0; i < 5; i++)
        cout << myVector.at(i) << endl;

    return 0;
}
```

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
```

```cpp
#include <iostream>
// Library required to use vectors
#include <vector>

using namespace std;

int main()
{
    // Initialize a vector with some values
    vector<int> myVector = {3, 1, 8};
    // Add elements to the end of vector
    myVector.push_back(5);
    myVector.push_back(2);
    // Iterate through all the elements
    for(int i = 0; i < 6; i++)
        cout << myVector.at(i) << endl;

    return 0;
}
```

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)
```

```cpp
#include <iostream>
// Library required to use vectors
#include <vector>

using namespace std;

int main()
{
    // Initialize a vector with some values
    vector<int> myVector = {3, 1, 8};
    // Add elements to the end of vector
    myVector.push_back(5);
    myVector.push_back(2);
    // Iterate through all the elements
    for(int i = 0; i < 6; i++)
        cout << myVector[i] << endl;

    return 0;
}
```

Delete an Element to a Vector: pop_back()

simple_vector.cpp

```
#include <iostream>
// Library required to use vectors
#include <vector>

using namespace std;

int main()
{
    // Initialize a vector with some values
    vector<int> myVector = {3, 1, 8};
    // Add elements to the end of vector
    myVector.push_back(5);
    myVector.push_back(2);
    // Delete the last element of vector
    myVector.pop_back();
    // Iterate through all the elements
    for(int i = 0; i < 4; i++)
        cout << myVector.at(i) << endl;

    return 0;
}
```

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

```
$ ./program
```

```
3
1
8
5
```

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

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

```
$ ./program
```

```
??
```

simple_vector.cpp

```
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

```
int main()
{
    // Initialize a vector with some values
    vector<int> myVector = {3, 1, 8};

    // Add and delete some elements
    myVector.push_back(5);
    myVector.push_back(2);
    myVector.pop_back();
    myVector.push_back(10);
    myVector.pop_back();
    myVector.pop_back();
    myVector.pop_back();

    // Get the number of elements present
    cout << myVector.size() << endl;
    return 0;
}
```

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

```
$ ./program
```

```
2
```

Take User Inputs to Put it in a Vector

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

```
$ ./program
```

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

user_inputs.cpp

```cpp
int main()
{
    // A vector of unsigned ints (0 and positives)
    vector<unsigned int> myVector;
    // An infinite loop to take user input
    // until the user inserts a negative value
    while(true)
    {
        int temp;
        cin >> temp;
        if(temp >= 0)
            myVector.push_back(temp);
        else
            break;
    }
    // Get the size of the vector
    int vectorSize = myVector.size();
    for(int i = 0; i < vectorSize; i++)
        cout << myVector[i] << endl;

    return 0;
}
```

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
```

```
int main()
{
    // A vector of unsigned ints (0 and positives)
    vector<unsigned int> myVector;
    // An infinite loop to take user input
    // until the user inserts a negative value
    while(true)
    {
        int temp;
        cin >> temp;
        if(temp >= 0)
            myVector.push_back(temp);
        else
            break;
    }
    // Get the size of the vector
    int vectorSize = myVector.size();
    for(int i = 0; i < vectorSize; i++)
        cout << myVector[i] << endl;

    return 0;
}
```

(Some of the) C++ Vector Functions

Function	Description
`size()`	returns the number of elements present in the vector
`clear()`	removes all the elements of the vector
`front()`	returns the first element of the vector
`back()`	returns the last element of the vector
`empty()`	returns **1** (true) if the vector is empty
`capacity()`	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

VARIABLES

Locals

myVector: size=0

vector_debug.cpp > main()

```cpp
int main()
{
    vector<int> myVector;
    myVector.push_back(1);
    myVector.push_back(2);
    myVector.push_back(3);
    myVector.push_back(4);
    myVector.pop_back();
    myVector.push_back(5);

    // Removes all the elements of the vector
    myVector.clear();

    return 0;
}
```

Vector: Debug Mode

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

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

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

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

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

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

VARIABLES

Locals

myVector: size=4

[0]: 1

[1]: 2

[2]: 3

[3]: 4

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

VARIABLES

Locals

myVector: size=3

[0]: 1

[1]: 2

[2]: 3

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

VARIABLES

Locals

myVector: size=4

[0]: 1

[1]: 2

[2]: 3

[3]: 5

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

VARIABLES

Locals

myVector: size=0

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: 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

??

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

??

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

??

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

```
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 }
```

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

```
$ ./program
```

```
100
200
300
400
500
```

for-each Loop for Vectors: Debug

VARIABLES

Locals

myVector: size=5

[0]: 100

[1]: 200

[2]: 300

[3]: 400

[4]: 500

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  }
```

for-each Loop for Vectors: Debug

VARIABLES

Locals

myVector: size=5

[0]: 100

[1]: 200

[2]: 300

[3]: 400

[4]: 500

item: 100

for_each_loop.cpp > main()

```cpp
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

VARIABLES

Locals

myVector: size=5

[0]: 100

[1]: 200

[2]: 300

[3]: 400

[4]: 500

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  }
```

for-each Loop for Vectors: Debug

VARIABLES

Locals

myVector: size=5

[0]: 100

[1]: 200

[2]: 300

[3]: 400

[4]: 500

item: 200

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  }
```

for-each Loop for Vectors: Debug

VARIABLES

Locals

myVector: size=5

[0]: 100

[1]: 200

[2]: 300

[3]: 400

[4]: 500

for_each_loop.cpp > main()

```cpp
int main()
{
    vector<int> myVector;
    for(int i = 1; i <= 5; i++)
        myVector.push_back(i * 100);

    // For each loop literals in C++
    for (int item : myVector)
        cout << item << endl;

    return 0;
}
```

for-each Loop for Vectors: Debug

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

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  }
```

for-each Loop for Vectors: Debug

VARIABLES

Locals

myVector: size=5

[0]: 100

[1]: 200

[2]: 300

[3]: 400

[4]: 500

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  }
```

for-each Loop for Vectors: Debug

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

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  }
```

for-each Loop for Vectors: Debug

VARIABLES

Locals

myVector: size=5

[0]: 100

[1]: 200

[2]: 300

[3]: 400

[4]: 500

for_each_loop.cpp > main()

```cpp
int main()
{
    vector<int> myVector;
    for(int i = 1; i <= 5; i++)
        myVector.push_back(i * 100);

    // For each loop literals in C++
    for (int item : myVector)
        cout << item << endl;

    return 0;
}
```

for-each Loop for Vectors: Debug

VARIABLES

Locals

myVector: size=5

[0]: 100

[1]: 200

[2]: 300

[3]: 400

[4]: 500

item: 500

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  }
```

for-each Loop for Vectors: Debug

VARIABLES

Locals

myVector: size=5

[0]: 100

[1]: 200

[2]: 300

[3]: 400

[4]: 500

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  }
```

Remarks

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