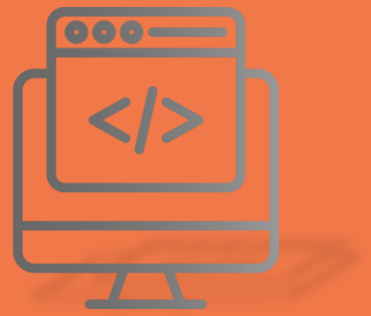
Tishk International University
Faculty of Applied Science
Information Technology Department

Vector and Deque



Programming II

Lecture 4

Spring 2026

Course Code: IT118

Grade 1

Islam Abdulazeez
islam.abdulaziz@tiu.edu.iq

May 31, 2026

Outlines

- ✓ Introduction to Vectors
- ✓ Defining and Initializing Vectors
- ✓ Vector Member Functions
- ✓ STL Algorithms with Vectors
- ✓ Copying and Passing Vectors to Functions
- ✓ Introduction to Deque
- ✓ Deque Operations
- ✓ Vector vs. Deque

Learning Outcomes

- **At the end of today's session, you will be able to:**

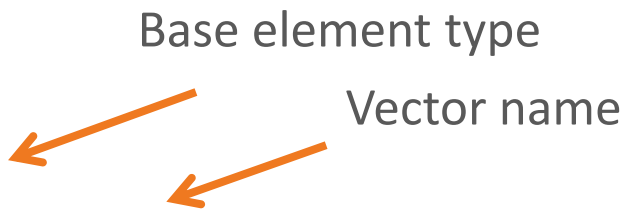
  - ✓ Define vectors and deques in C++.
  - ✓ Explain the features and advantages of vectors and deques.
  - ✓ Use vector and deque functions to manipulate data.
  - ✓ Compare between vectors and deques.

Introduction to Vector

- Arrays are used to store sequential data and are static in nature. Generally, arrays are non-dynamic (static), meaning they have a fixed size. However, C++ also provides **dynamic arrays**, known as **vectors**, which can grow or shrink as needed.
- Vectors can automatically resize themselves when elements are inserted or deleted, depending on the requirements of the task being executed. This is not the same as an array, where only a fixed number of values can be stored under a single variable name.
- Vector is a dynamic array container provided by the Standard Template Library (STL) in C++.

Defining a New Vector

- Basic construction




```
vector<int> A;

vector<int> A;        // 0 ints

vector<float> B;      // 0 floats

vector<int> - vector of integers.
vector<string> - vector of strings.
vector<int * > - vector of pointers to integers.
```

Defining a New Vector

```
#include<vector>
```

- Declaration: vector<type> vectorName(size);

  Example: vector<int> a(3);
  Example: vector<int> b; // Zero size
  Example: vector<int> A(5); // 5 ints
  Example: vector<float> B(10); // 10 floats

- Vector can be indexed as an array, using [ ]

Defining a New Vector

vector<type> vectorName(size, value);

- Creates a vector with a specified number of elements (size), where each element is initialized with the given value.

```
vector<int> a(3, 5);
```

vector<float> b(20,1.5);

- Creates a vector named b containing 20 elements, and each element has the value 1.5.Result: {1.5, 1.5, 1.5, ..., 1.5} (20 times)

Defining a New Vector

- In the below example, a blank vector is being created. Vector is a dynamic array and, doesn't need size declaration.

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<int> my_vector;
    vector<int> A = {1,2,6,-9,5};
    vector<int> C {1,2,6,-9,5};
}
```

Why Vector?

- Provides an alternative to the built in array.
- Vector is self grown.
- Use it instead of the built in array!
- It is one of the most commonly used data structures in C++.
- Vector provides a dynamic, resizable array-like container with several built-in functionalities

Vector member functions

v{7,3,4,0,9,-4,-3,5,2,11}

- v.size(); // return the number of elements in v , which is 10
- v.push_back(14); //appends 14 to the end of v{7,3,4,0,9,-4,-3,5,2,11,14};
- v.at(4); // retrieves the element at index 4, which is 9
- v.resize(15) // change size of v to 15
- v.pop_back() // remove the last element in v {7,3,4,0,9,-4,-3,5,2};
- v.front(); // return the first element in v, which is 7
- v.back(); // return the last element in v, which is 11
- v.clear(); // delete all the element in v
- v.empty(); // return 1 if v is empty; else return 0;
- v.begin(); //the beginning of the vector, which is 7
- v.erase(); // erases a specific element

Vectors member functions: Size()

- Returns the number of elements in the vector.

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6      vector<int> P={1,4,0,8,6,-4};
7      cout<<P.size()<<endl;
8  }
```

```
/tmp/491WVkpYmS.o
6
```

Vectors size vs array size

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int A[3]={1,4,6}; // Array
    vector<int> V = {1,5,5,2}; // Vector
    cout<<"Array Size "<<sizeof(A)/sizeof(A[0])<<endl;
    cout<<"Vector Size "<<V.size()<<endl;

    return 0;
}
```

Example of size()

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6      vector<int> P={1,4,0,8,6,-4};
7      for(int i=0;i<P.size();i++){
8          cout<<P[i]<<"\t";
9      }
10 }
```

```
/tmp/491WVkpYmS.o
1    4    0    8    6    -4
```

Vectors member functions: Adding elements

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

int main() {
    vector<int> A;
    A.push_back(2); // Adding 2 to vector
    cout<<"Adding 2 to an empty vector A"<<endl;
    for(int i=0;i<A.size();i++){
        cout<<A[i]<<"\t";
    }


    A.push_back(3); // Adding 3 to vector A
    for(int i=0;i<A.size();i++){
        cout<<A[i]<<"\t";
    }
}
```

```
/tmp/491WVkpYmS.o
Adding 2 to an empty vector A
2
Adding 2 to vecote A
2    3
```

Adding numbers by For loop

- Use a for loop to iteratively input values into the vector.

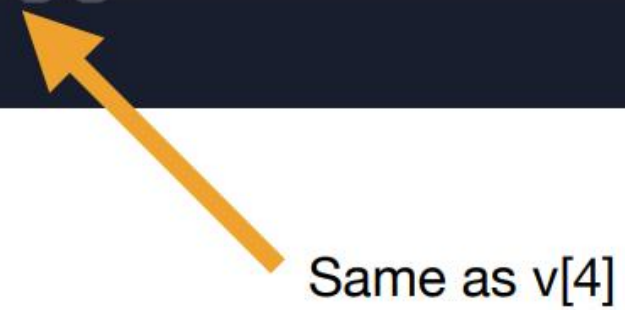
```
#include <iostream>
using namespace std;
#include<vector>
int main() {
    vector<int> V;
    int x;
for(int i=0;i<5;i++){
    cout<<"Input number"<<endl;
    cin>>x;
    V.push_back(x);
}

for(int i=0;i<5;i++){
    cout<<V[i]<<endl;
}
    return 0;
}
```

Vectors member functions: at()

- **at**(index): retrieves the element at index 4, which is 9

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6          vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
7          cout<<v.at(4);
8  }
```

```
/tmp/491WVkpYmS.o
9
```



Vectors member functions: resize()

- **resize**(newSize): Resizing a vector to a new size.

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

int main() {
        vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
        v.resize(12);
        for(int i=0;i<v.size();i++){
            cout<<v[i]<<" ";
        }
}
```

```
/tmp/491WVkpYmS.o
7 3 4 0 9 -4 -3 5 2 11 0 0
```

Vectors member functions: pop_back()

- **v.pop_back()**: Removes the last element of the vector

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6          vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
7          v.pop_back();
8          for(int i=0;i<v.size();i++){
9              cout<<v[i]<<" ";
10         }
11 }
```

```
/tmp/491WVkpYmS.o
7 3 4 0 9 -4 -3 5 2
```

Vectors member functions: front()

- **v.front()**: return the first element in v, which is 7

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6          vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
7          cout<<v.front();
8  }
```

```
/tmp/
7
```

Vectors member functions: back()

- v.**back()**: Returns the last element in the vector

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

int main() {
    vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
    cout<<v.back()<<endl;

}
```

```
/tm
11
```

Vectors member functions: clear() & empty()

- v.**clear()**: Removes all elements inside the vector
- v.**empty()**: Check the empty vector, return 1 if vector is empty; else return 0

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6      vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
7      cout<<v.empty()<<endl;
8      v.clear();
9      cout<<v.empty()<<endl;
10 }
```

```
/tmp
0
1
```

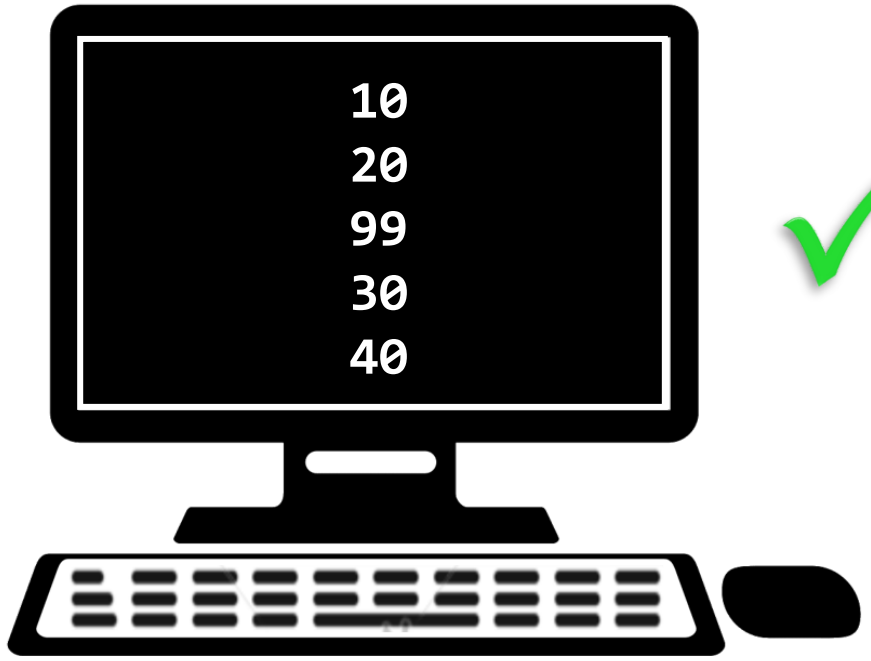
Vectors member functions: insert()

- The **insert()** function is used to add an element at a specific position inside a vector.

  vector_name.insert(position, value);

- Example

```
vector<int> v = { 10, 20, 30, 40 };

v.insert(v.begin() + 2, 99);

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



Vectors member functions: erase()

- v.**erase():** It is used to remove elements from a vector at a specified position or within a range.

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

int main() {
    vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
    v.erase(v.begin());
    for(int i=0;i<v.size();i++){
        cout<<v[i]<<" ";
    }

}
```

```
/tmp/ZEov6iQKAs.o
3 4 0 9 -4 -3 5 2 11
```

Vectors member functions: erase()

- Removes the range between index 2 until 4

```
1  #include <iostream>                                  /tmp/ZEov6iQKAs.o
2  #include <vector>                                    7 3 9 -4 -3 5 2 11
3  using namespace std;
4
5  int main() {
6      vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
7      v.erase(v.begin()+2, v.begin()+4);
8      for(int i=0;i<v.size();i++){
9          cout<<v[i]<<" ";
10     }
11 }
```

STL Algorithms with Vectors:

- The STL (Standard Template Library) provides a set of algorithms that work seamlessly with vectors such as **sort()**, **accumulate()** and **…** .

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  int main() {
7      vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
8      sort(v.begin(), v.end());
9      for(int i=0;i<v.size();i++){
10         cout<<v[i]<<" ";
11     }
12 }
```

```
/tmp/ZEov6iQKAs.o
-4 -3 0 2 3 4 5 7 9 11
```

STL Algorithms with Vectors:

- **accumulate()**: It is used to perform a summation or accumulation operation on a range of elements

sum variable starting from the initial value of 0.

```
1  #include <iostream>                                    /tmp/Z
2  #include <vector>                                      34
3  #include <numeric>
4  using namespace std;
5
6  int main() {
7      vector<int> v={7,3,4,0,9,-4,-3,5,2,11};
8      int x = accumulate(v.begin(), v.end(),0);
9      cout<<x;
10 }
```

Copies One Vector Into Another

```cpp
vector<int> v1 = { 1, 2, 3, 4, 5 };
vector<int> v2;

// Copy vector
v2 = v1;

// Print copied vector
for (int i = 0; i < v2.size(); i++) {
    cout << v2.at(i) << " ";
}
```

Vector and functions

- By using function, write a program that finds the largest number inside a vector.

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

int vecLarge(vector<int> A){
    int largest = A[0];
    for (int i=1;i<A.size();i++){
        if(A[i]>largest){
            largest = A[i];
        }
    }
    return largest;
}
int main() {
    vector<int> V = {1,-4,16,7,-9};
    cout<<vecLarge(V)<<endl;

    return 0;
}
```

Deque

- **Double Ended Queue**
- Functionality similar to vectors, but with efficient insertion and deletion of elements also at the beginning of the sequence, and not only at its end.
- Same basic functions as vector, in addition to that deque supports **push_front** and **pop_front** for insertion and deletion at beginning of deque.

Deque

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

int main() {
    deque<int> d;

    // Insert elements at the front
    d.push_front(10);
    d.push_front(20);
    d.push_front(30);

    cout << "Deque after push_front operations: ";
    for (int i = 0; i < d.size(); i++) {
        cout << d.at(i) << " ";
    }
    cout << endl;

    // Remove elements from the front
    d.pop_front();
    d.pop_front();

    cout << "Deque after pop_front operations: ";
    for (int i = 0; i < d.size(); i++) {
        cout << d.at(i) << " ";
    }
    return 0;
}
```

Deque Library

Output:

Deque after push_front operations: 30 20 10

Deque after pop_front operations: 10

Activities and Next Lecture's Topic

Activities

- Review this lecture note
- Practice

Next Lecture's Topic

- File Stream

References

- Gaddis, T. (2014). Starting out with C++: Early objects (7th ed.). Pearson Education.

Thank You!