Vector in

01 Example

02 Importance

03 Data Manipulation Operations

04 Functions Correlated

By – Mohammad Imran

Vector

Introduction

Vectors in C++ are one of the containers offered to us by the STL (Standard Template Library) in C++. It stores a collection of similar-type objects in a variable-sized array.

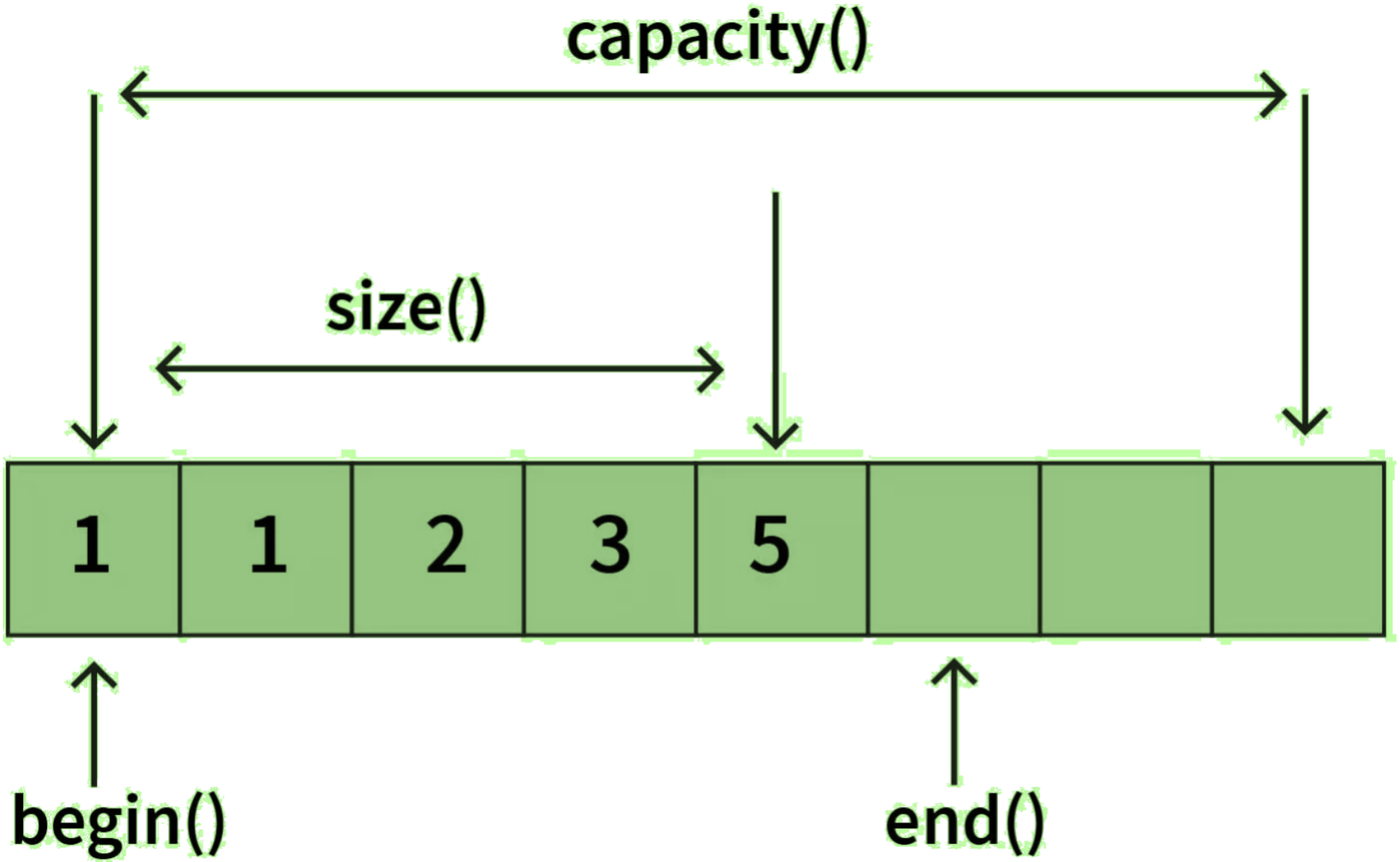
A vector is part of the Standard Template Library (STL) and is a sequence container that can dynamically resize itself to accommodate elements. It is similar to arrays but offers more flexibility.

Note: C++ provides us with objects that store a collection of elements (i.e. other objects), referred to by the term ‘containers’.

By – Mohammad Imran

Vector

Introduction



By – Mohammad Imran

Vector Declaration

A vector is a dynamic array, meaning that its size can grow or shrink as needed during program execution. Vectors are part of the C++ Standard Template Library (STL)

Syntax:

```
std::vector <data_type> vector_name;
```

std::vector is the namespace for vectors

<data_type> is the type of data the vector will store, such as int, double.

vector_name is the name of the vector

By – Mohammad Imran

Vector Initialization

- ✓ **Initialization Using List**

Example:

```
std::vector<int> numbers = {1, 2, 3, 4, 5};
```

- ✓ **Initialization with Single Value**

Example:

```
std::vector<int> numbers(10, 0);
Initialize a vector of 10 elements with all values set to
0.
```

By – Mohammad Imran

Vector Initialization

✓ **Initialization Using Copy Constructor**

Example:

```
std::vector<int> numbers1 = {1, 2, 3};
std::vector<int> numbers2(numbers1);
```

Note: Copy the values of numbers1 to numbers2

Vector Initialization

- ✓ **Initialization Using an Array**

Example:

```
int arr[] = {1, 2, 3};
std::vector<int>numbers(arr, arr + sizeof(arr)/sizeof(int));
```

Note: Initialize a vector using the array arr

Vector

Basic Operation

✓ Accessing Element

You can access elements of a vector using either its index or an iterator. Indexing uses square brackets [] and an index value, while iterators provide a pointer-like interface to traverse the vector.

You can also access element of a vector using **at()** function.

Vector

Example - 1

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector <int> vr = {10, 20, 30, 40, 50};
    cout << vr[0] << endl;
    cout << vr.at(0);
    return 0;
}
```

OUTPUT

```
10
10
```

By – Mohammad Imran

Vector

Basic Operation

✓ Adding Element

There are several ways to add new elements to a vector. The most common methods are –

- `push_back()`
- `insert()`

Vector

Example - 2

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   vector <int> vr;
   int num;
   for(int i=1; i<6; i++)
   {
      cout << "Enter Element : ";
      cin >> num;
      vr.push_back(num);
   }
   for (int i=0; i<6; ++i)
   {
      cout << vr[i] << " ";
   }
   return 0;
}
```

By – Mohammad Imran

✓ Removing Element

You can remove elements from a vector using the given function –

- ✓ `pop_back()`
- ✓ `erase()`

Vector

Example - 3

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   vector <int> vr;
   int num;
   for(int i=1; i<6; i++)
   {
      cin >> num;
      vr.push_back(num);
   }
   for (int i=0; i<6; ++i)
   {
      cout << vr[i] << " ";
   }
   cout << endl;
   vr.pop_back();
   for (int i=0; i<6; ++i)
   {
      cout << vr[i] << " ";
   }

   return 0;
}
```

By – Mohammad Imran

Vector

Basic Operation

✓ Checking Size and Emptiness

The **size()** function returns the number of elements in the vector, while the **empty()** function checks whether the vector is empty or not.

Vector

Example - 4

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   vector <int> vr = {1, 2, 3, 4, 5};
   vector <int> emp;
   int n1 = vr.size();
   int n2 = emp.size();
   cout << "Size of vr = " << n1 << endl;
   cout << "Size of emp = " << n2 << endl;
   bool b1 = vr.empty();
   bool b2 = emp.empty();
   cout << "vr = " << b1 << endl;
   cout << "emp = " << b2;
   return 0;
}
```

OUTPUT

```
Size of vr = 5
Size of emp = 0
vr = 0
emp = 1
```

By – Mohammad Imran

Vector — Various Functions — Capacity

Function	Description	Example Code
size()	Return size	size_t size = a.size();
max_size()	Return maximum size	size_t maxSize = a.max_size();
resize(n)	Change size	a.resize(10);
capacity()	Return size of allocated storage capacity	size_t capacity = a.capacity();
empty()	Test whether vector is empty	bool isEmpty = a.empty();
reserve()	Request a change in capacity	a.reserve(100);
shrink_to_fit()	Shrink to fit	a.shrink_to_fit();

By – Mohammad Imran

Vector | Capacity Function | Example - 5

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<int> a;
    for (int i = 1; i <= 5; i++)
        a.push_back(i);
    cout << "Size : " << a.size();
    cout << "\nCapacity : " << a.capacity();
    cout << "\nMax_Size : " << a.max_size();
    a.resize(4);
    cout << "\nSize : " << a.size();
    if (!a.empty())
        cout << "\nVector is not empty";
    else
        cout << "\nVector is empty";
    return 0;
}
```

OUTPUT

```
Size : 5
Capacity : 8
Max_Size : 4611686018427
                    387903
Size : 4
Vector is not empty
```

By – Mohammad Imran

Vector

Various Functions

Modifiers

Function	Description	Example Code
assign()	Assign vector content	a.assign(4, 7);
push_back()	Add element at the end	a.push_back(10);
pop_back()	Delete last element	a.pop_back();
insert()	Insert elements	a.insert(a.begin(), 3);
erase()	Erase elements	a.erase(a.begin());
swap()	Swap content	a.swap(b);
clear()	Clear content	a.clear();
emplace()	Construct and insert element	a.emplace(a.begin(), 42);
emplace_back()	Construct and insert element at the end	a.emplace_back(42);

By – Mohammad Imran

Vector | Modifier Function | Example - 6

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   vector<int> a;
   vector<int> a1, a2;
   a.assign(4, 7);
   cout << "The vector contains: ";
   for (int i = 0; i < a.size(); i++)
      cout << a[i] << " ";
   cout << endl;
   a.push_back(10);
   int n = a.size();
```

By – Mohammad Imran

```
cout << "The last element is: " << a[n - 1] << endl;
a.pop_back();
cout << "The vector contains: ";
for (int i = 0; i < a.size(); i++)
   cout << a[i] << " ";
cout << endl;
a.insert(a.begin(), 3);
cout << "The first element is: " << a[0] << endl;
a.erase(a.begin());
cout << "The first element is: " << a[0] << endl;
a.clear();
cout << "Vector size after erase(): " << a.size() << endl;
a1.push_back(3);
a1.push_back(4);
a2.push_back(5);
```

Vector | Modifier Function | Example - 6

```
    a2.push_back(6);
    cout << "\nVector 1 is: ";
    for (int i = 0; i < a1.size(); i++)
       cout << a1[i] << " ";
    cout << "\nVector 2 is: ";
    for (int i = 0; i < a2.size(); i++)
       cout << a2[i] << " ";
    a1.swap(a2);
    cout << "\nAfter Swap \nVector 1 is: ";
    for (int i = 0; i < a1.size(); i++)
       cout << a1[i] << " ";
    cout << "\nVector 2 is: ";
    for (int i = 0; i < a2.size(); i++)
       cout << a2[i] << " ";
    return 0;
}
```

OUTPUT

```
Vector contains:7 7 7 7
Last element is:10
Vector contains:7 7 7 7
First element is:3
First element is:7
Size after erase():0

Vector 1 is:3 4
Vector 2 is:5 6
After Swap
Vector 1 is:5 6
Vector 2 is:3 4
```

By – Mohammad Imran

Vector

Various Functions

Element Access

Function	Description	Example Code
operator[]	Access element	int element = a[0];
at()	Access element with bounds checking	int element = a.at(0);
front()	Access first element	int firstElement = a.front();
back()	Access last element	int lastElement = a.back();
data()	Access data as a pointer	int* dataPtr = a.data();

By – Mohammad Imran

Vector | Element Access Function | Example - 7

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   vector<int> a;
   for (int i = 1; i <= 10; i++)
   a.push_back(i * 10);
   cout << "\nReference operator [g] : a[2] = " << a[2];
   cout << "\nat : a.at(4) = " << a.at(4);
   cout << "\nfront() : a.front() = " << a.front();
   cout << "\nback() : a.back() = " << a.back();
   int* pos = a.data();
   cout << "\nThe first element is " << *pos;
   return 0;
}
```

OUTPUT

```
Reference operator [g]:a[2]=30
at : a.at(4) = 50
front() : a.front() = 10
back() : a.back() = 100
The first element is 10
```

By – Mohammad Imran