**Array**

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100; // Maximum size of the array

// Function to insert an element at the end of the array

void insert(int arr[], int &size, int element)

{

if (size < MAX\_SIZE)

{

arr[size++] = element;

cout << "Element inserted successfully.\n";

}

else

{

cout << "Array is full. Cannot insert element.\n";

}

}

// Function to display all elements of the array

void display(const int arr[], int size)

{

cout << "Array elements: ";

for (int i = 0; i < size; ++i)

{

cout << arr[i] << " ";

}

cout << "\n";

}

// Function to search for an element in the array

int search(const int arr[], int size, int element)

{

for (int i = 0; i < size; ++i)

{

if (arr[i] == element)

{

return i; // Return index of the element if found

}

}

return -1; // Return -1 if element not found

}

// Function to delete an element from the array

void remove(int arr[], int &size, int element)

{

int index = search(arr, size, element);

if (index != -1)

{

// Shift elements to the left to fill the gap

for (int i = index; i < size - 1; ++i)

{

arr[i] = arr[i + 1];

}

size--; // Reduce the size of the array

cout << "Element removed successfully.\n";

}

else

{

cout << "Element not found in the array.\n";

}

}

int main()

{

int arr[MAX\_SIZE];

int size = 0;

insert(arr, size, 5);

insert(arr, size, 10);

insert(arr, size, 15);

display(arr, size); // Output: Array elements: 5 10 15

cout << "Index of 10: " << search(arr, size, 10) << "\n"; // Output: Index of 10: 1

remove(arr, size, 10);

display(arr, size); // Output: Array elements: 5 15

return 0;

}

**STD::Array**

#include <algorithm>

#include <array>

#include <iostream>

#include <iterator>

#include <string>

using namespace std;

int main()

{

// construction uses aggregate initialization

// double-braces required

array<int, 5> ar1{{3, 4, 5, 1, 2}};

array<int, 5> ar2 = {1, 2, 3, 4, 5};

array<string, 2> ar3 = {{string("a"), "b"}};

cout << "Sizes of arrays are" << endl;

cout << ar1.size() << endl;

cout << ar2.size() << endl;

cout << ar3.size() << endl;

cout << "\nInitial ar1 : ";

for (auto i : ar1)

cout << i << ' ';

// container operations are supported

sort(ar1.begin(), ar1.end());

cout << "\nsorted ar1 : ";

for (auto i : ar1)

cout << i << ' ';

// Filling ar2 with 10

ar2.fill(10);

cout << "\nFilled ar2 : ";

for (auto i : ar2)

cout << i << ' ';

// ranged for loop is supported

cout << "\nar3 : ";

for (auto &s : ar3)

cout << s << ' ';

// [ ] operator

array <char , 3> arr={'G','f','G'};

cout<<arr[0] <<" "<<arr[2];

// front( ) and back( )

cout<<arr.front() <<" "<<arr.back();

// swap( )

arr2.swap(arr1);

cout<<arr.front() <<" "<<arr.back();

// empty()

bool x = arr.empty(); // false ( not empty)

cout<<boolalpha<<(x);

// at()

cout<< arr.at(2) <<" " << arr1.at(2);

// fill()

arr.fill(1);

for(int i: arr)

cout<<arr[i]<<" ";

// size( ) or max\_size( ) and sizeof( ) function

cout<<arr.size()<<'\n'; // total num of indexes

cout<<arr.max\_size()<<'\n'; // total num of indexes

cout<<sizeof(arr); // total size of array

// data()

const char\* str = "GeeksforGeeks";

array<char,13> arr;

memcpy (arr.data(),str,13);

cout << arr.data() << '\n';

return 0;

}

**STL::Vector**

#include <iostream>

#include <vector>

using namespace std;

int main()

{

////////////////// Iterators ////////////////

vector<int> g1;

for (int i = 1; i <= 5; i++)

g1.push\_back(i);

cout << "Output of begin and end: ";

for (auto i = g1.begin(); i != g1.end(); ++i)

cout << \*i << " ";

cout << "\nOutput of cbegin and cend: ";

for (auto i = g1.cbegin(); i != g1.cend(); ++i)

cout << \*i << " ";

cout << "\nOutput of rbegin and rend: ";

for (auto ir = g1.rbegin(); ir != g1.rend(); ++ir)

cout << \*ir << " ";

cout << "\nOutput of crbegin and crend : ";

for (auto ir = g1.crbegin(); ir != g1.crend(); ++ir)

cout << \*ir << " ";

//////////////////// Capacity /////////////////////

vector<int> g1;

for (int i = 1; i <= 5; i++)

g1.push\_back(i);

cout << "Size : " << g1.size();

cout << "\nCapacity : " << g1.capacity();

cout << "\nMax\_Size : " << g1.max\_size();

// resizes the vector size to 4

g1.resize(4);

// prints the vector size after resize()

cout << "\nSize : " << g1.size();

// checks if the vector is empty or not

if (g1.empty() == false)

cout << "\nVector is not empty";

else

cout << "\nVector is empty";

// Shrinks the vector

g1.shrink\_to\_fit();

cout << "\nVector elements are: ";

for (auto it = g1.begin(); it != g1.end(); it++)

cout << \*it << " ";

/////////////////////// Element Access ///////////////////////

vector<int> g1;

for (int i = 1; i <= 10; i++)

g1.push\_back(i \* 10);

cout << "\nReference operator [g] : g1[2] = " << g1[2];

cout << "\nat : g1.at(4) = " << g1.at(4);

cout << "\nfront() : g1.front() = " << g1.front();

cout << "\nback() : g1.back() = " << g1.back();

// pointer to the first element

int\* pos = g1.data();

cout << "\nThe first element is " << \*pos;

////////////////////// Modifiers ///////////////////////////

// Assign vector

vector<int> v;

// fill the vector with 10 five times

v.assign(5, 10);

cout << "The vector elements are: ";

for (int i = 0; i < v.size(); i++)

cout << v[i] << " ";

// inserts 15 to the last position

v.push\_back(15);

int n = v.size();

cout << "\nThe last element is: " << v[n - 1];

// removes last element

v.pop\_back();

// prints the vector

cout << "\nThe vector elements are: ";

for (int i = 0; i < v.size(); i++)

cout << v[i] << " ";

// inserts 5 at the beginning

v.insert(v.begin(), 5);

cout << "\nThe first element is: " << v[0];

// removes the first element

v.erase(v.begin());

cout << "\nThe first element is: " << v[0];

// inserts at the beginning

v.emplace(v.begin(), 5);

cout << "\nThe first element is: " << v[0];

// Inserts 20 at the end

v.emplace\_back(20);

n = v.size();

cout << "\nThe last element is: " << v[n - 1];

// erases the vector

v.clear();

cout << "\nVector size after clear(): " << v.size();

// two vector to perform swap

vector<int> v1, v2;

v1.push\_back(1);

v1.push\_back(2);

v2.push\_back(3);

v2.push\_back(4);

cout << "\n\nVector 1: ";

for (int i = 0; i < v1.size(); i++)

cout << v1[i] << " ";

cout << "\nVector 2: ";

for (int i = 0; i < v2.size(); i++)

cout << v2[i] << " ";

// Swaps v1 and v2

v1.swap(v2);

cout << "\nAfter Swap \nVector 1: ";

for (int i = 0; i < v1.size(); i++)

cout << v1[i] << " ";

cout << "\nVector 2: ";

for (int i = 0; i < v2.size(); i++)

cout << v2[i] << " ";

return 0;

}