**Vector**:

Vector are sequence containers representing arrays that can change in size. Vector store elements in contiguous storage location for their elements.

**Properties:**

Sequence

Elements in sequence container are ordered in strict linear sequence, individual elements are accessed based on the position in the sequence.

Dynamic array

Allows direct access to any elements in sequence and provides fast add/remove of elements at the end of the sequence.

Allocator-aware

Allocator object is used to dynamically handle its storage needs.

**Initializing vector:**

NOTE: compile with c++11 std recomended

Method-1:

Data can be inserted to vector by push\_back mechanism.

Program:

#include<iostream>

#include<vector>

int main()

{

std::vector<int> vtr;

vtr.push\_back(10);

vtr.push\_back(20);

vtr.push\_back(30);

for(int x: vtr)

std::cout << x << " ";

std::cout << std::endl;

return 0;

}

Output:

10 20 30

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Method-2:

Specifing the size and initializing the data.

syntax

std::vector<data\_type> vector\_name (size, value)

This vector has same value throught its elements.

Program:

#include<iostream>

#include<vector>

int main()

{

int n = 3;

std::vector<int> vtr(n, 11);

//this for loop works based on range

for(int x: vtr)

std::cout << x << " ";

std::cout << std::endl;

return 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Method-3:

Initializing like array

Program:

#include<iostream>

#include<vector>

int main()

{

std::vector<int> vtr {10, 20, 30};

for(int x: vtr)

std::cout << x << " ";

std::cout << std::endl;

return 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Method-4:

Initializing from array

Program:

#include<iostream>

#include<vector>

int main()

{

int arr[] = {100, 200, 300};

int size = sizeof(arr) / sizeof(arr[0]);

//vtr will interate from arr[0] o aray max size

std::vector<int> vtr(arr, arr + size);

for(int x: vtr)

std::cout << x << " ";

std::cout << std::endl;

return 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Method-5:

Initialize from another vector

Program:

#include<iostream>

#include<vector>

int main()

{

std::vector<int> v1 = {20, 40, 60};

//v1.begin will point to 20

//v1.end will point to 60

// vtr will iterate from v1 begin to v1 end

std::vector<int> vtr(v1.begin(), v1.end());

for(int x: vtr)

std::cout << x << " ";

std::cout << std::endl;

return 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Important Terms:**

Size -> It denotes the number of elements in the vector

Capacity -> It denoted the range of elements the vector can hold, once a particular range is filled. Vector will allocate more range.

push\_back -> It will increase the size of elements by one value by inserting the value to vector at end of it.

Reserve -> It reserve the minimum memory for the vector.

**How does vector dynamically increase its size?**

Capacity = 2 power N

where N start from 0.

Initially the vector capacity is 1.

When N = 1

The capacity of vector increases to 2

When N = 2

The capacity of vector increases to 4

When N = 3

The capacity of vector increases to 8

and so on.

Normally if we know that vector minimum require 50 capacity we can avoid the dynamic increase in size by reserving the minimum memory by using,

vector\_name.reserve(50)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note:

When capacity is 2 ,size is 2 and user try to insert a new element, The capacity increase to 4 and then a new contiguos memory is selected by the compiler and those 2 data previously present are copied to the newly allocated memory and then the third element is added to the vector.