**[6] - STL - Vector**

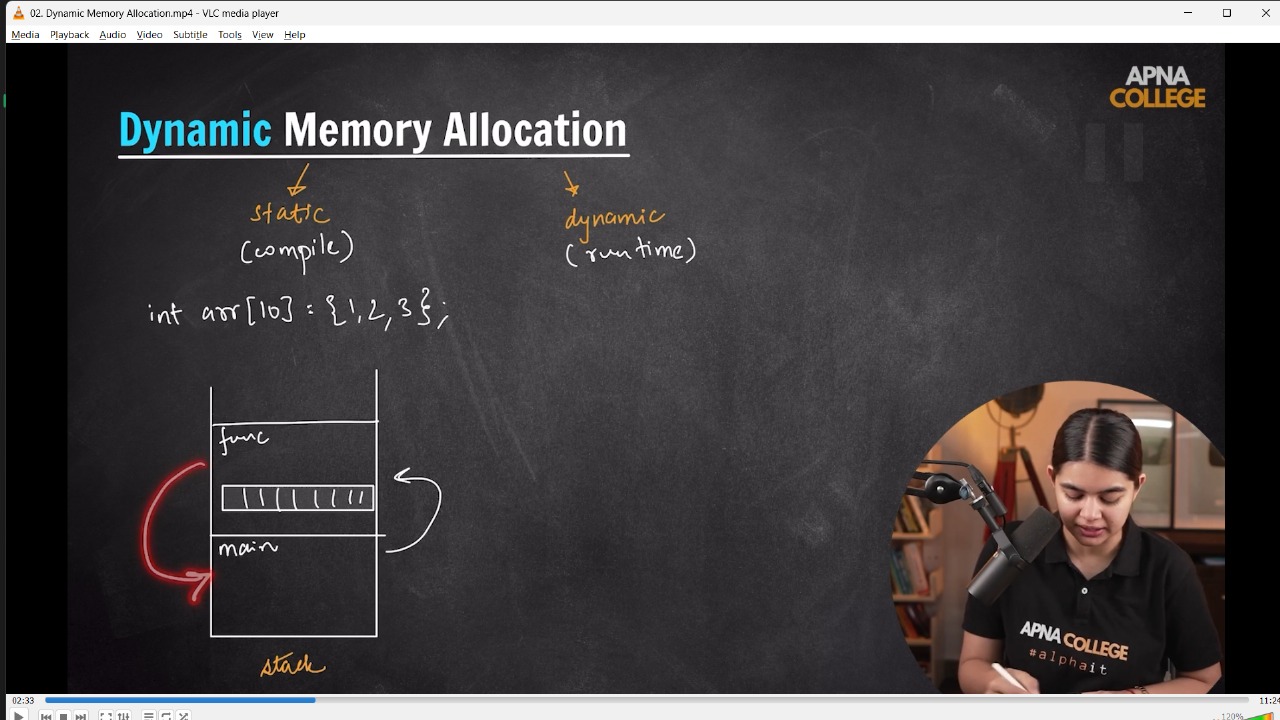
1) Pair\_Sum - Find if any pair in Sorted Array has target sum

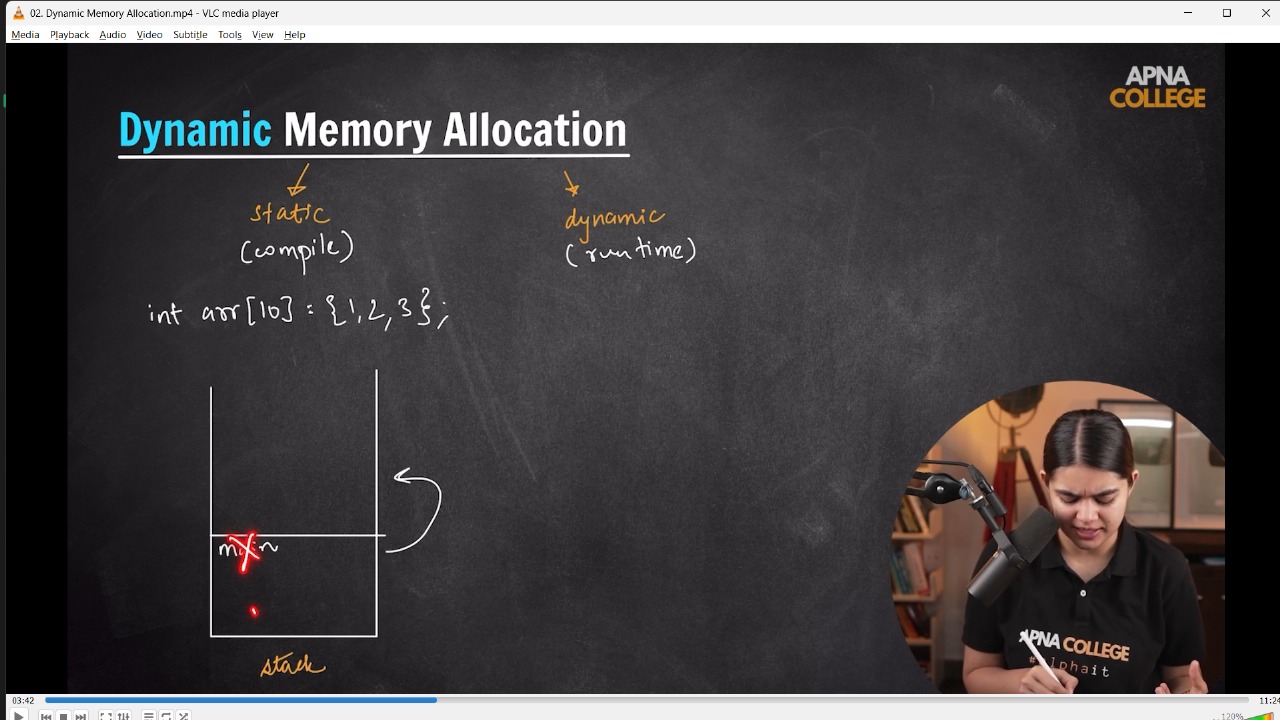
1.1) Brute Force Approach - using Nested Loops. TC - O(n^2)

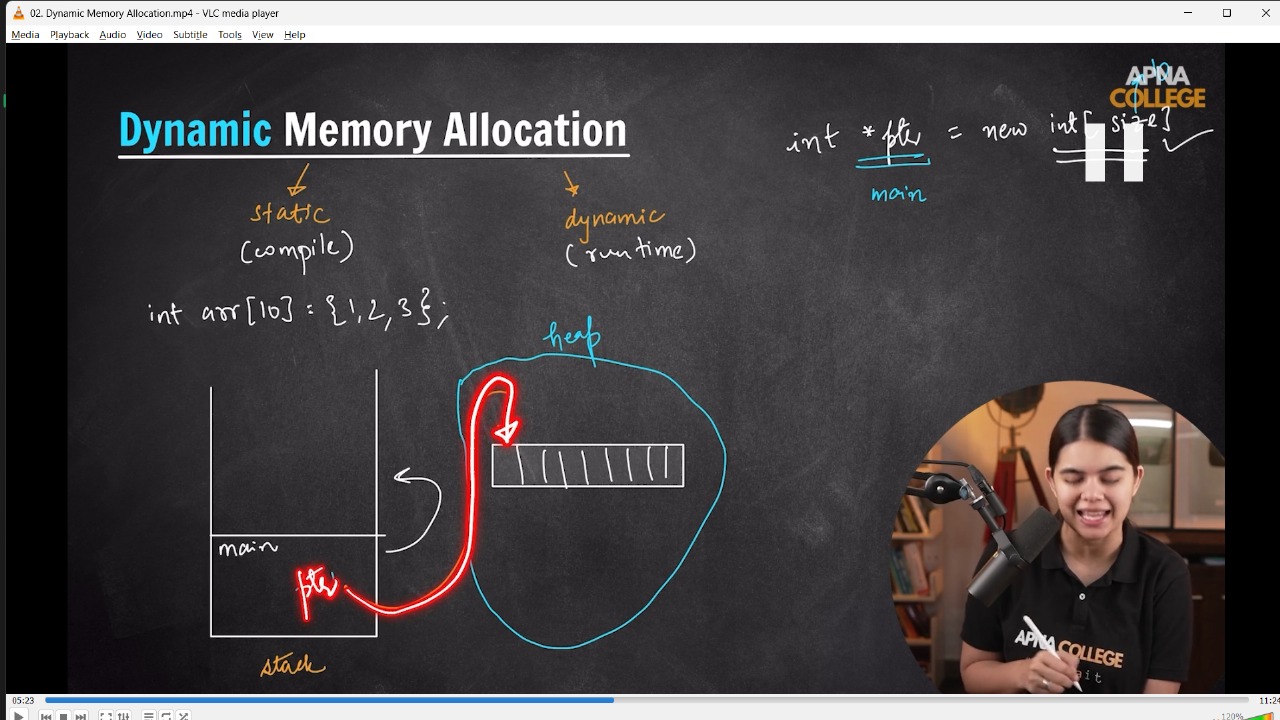
1.3) Using Linear Approach - 2 Pointer Approach. TC - O(n)

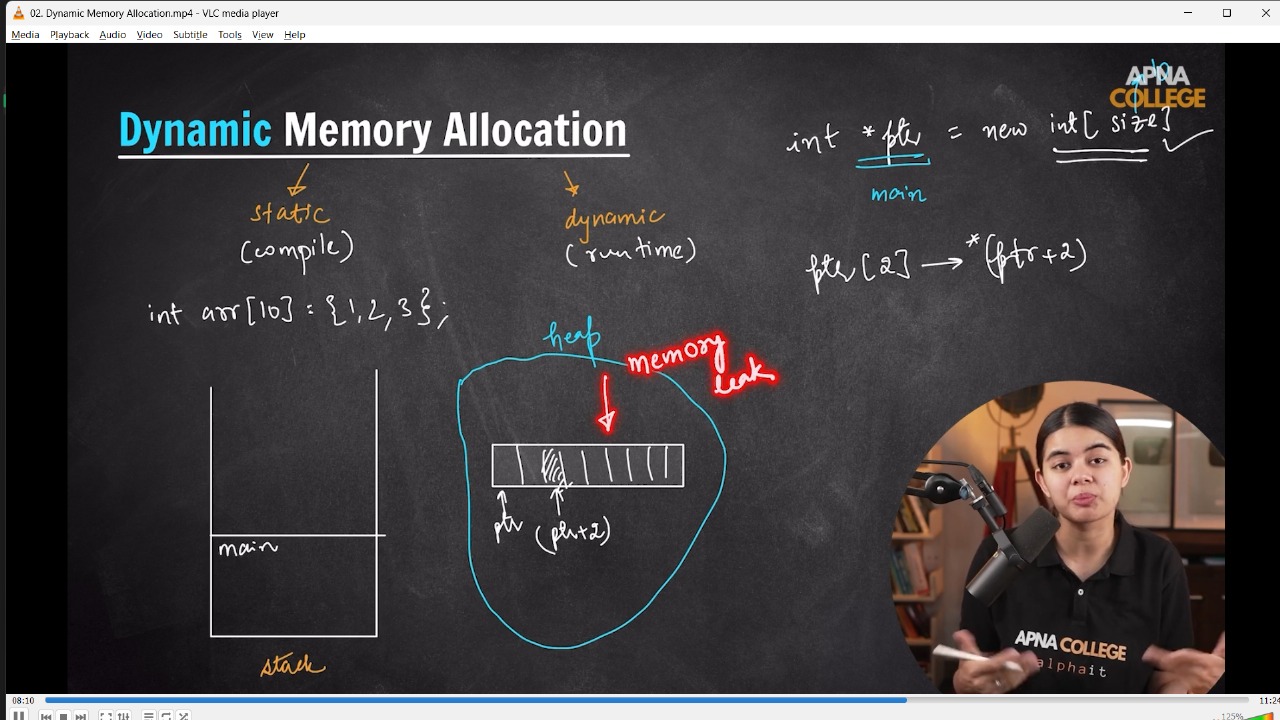
**6) DMA & STL in CPP –**

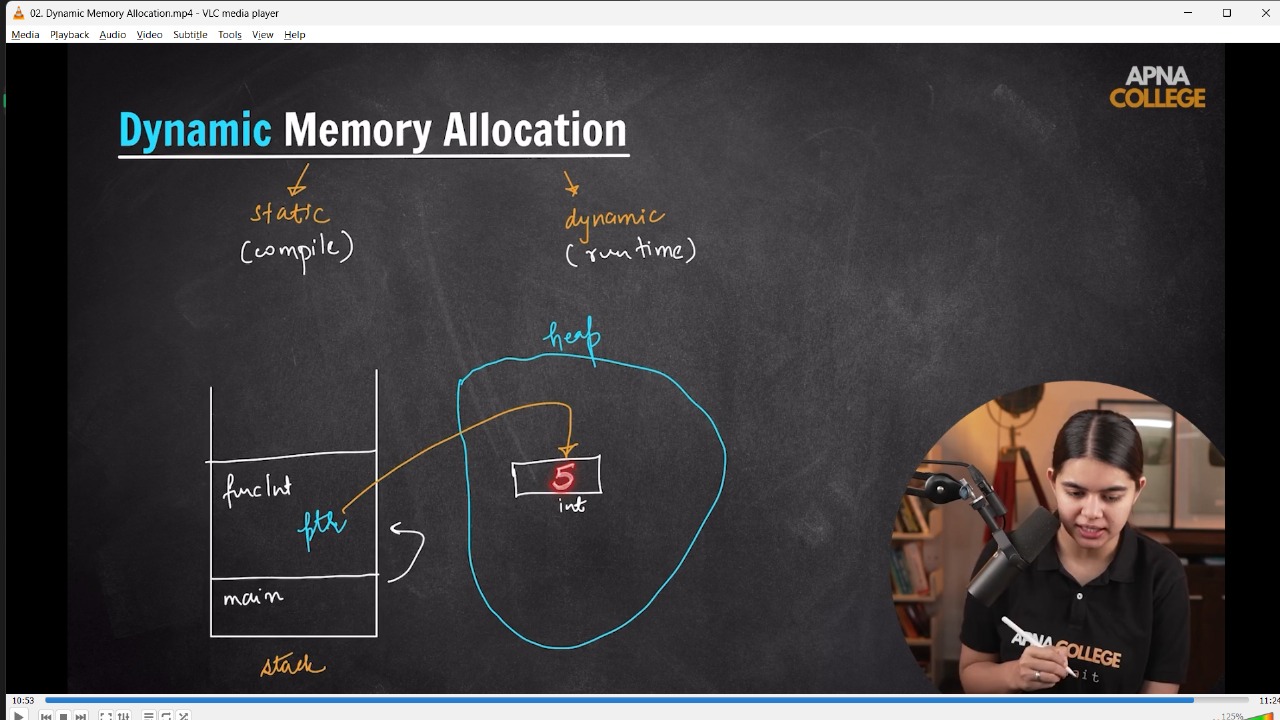
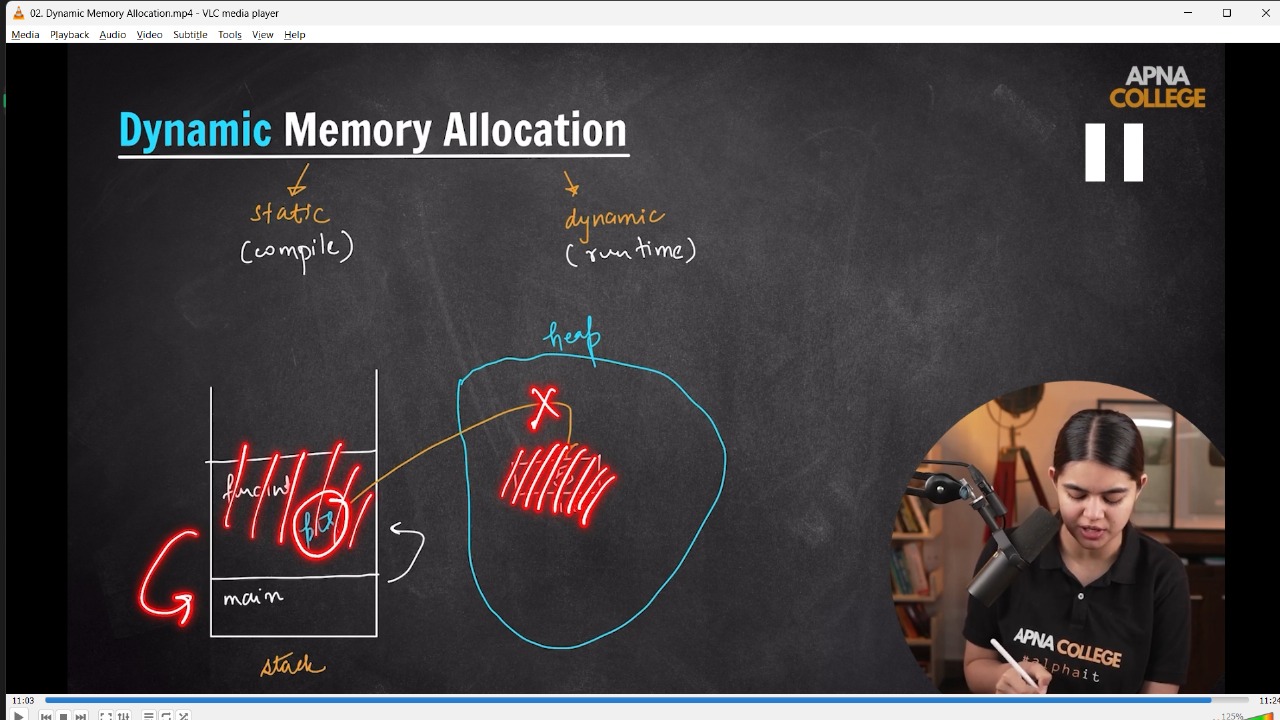
**//1) Dynamic Memory Allocation -**



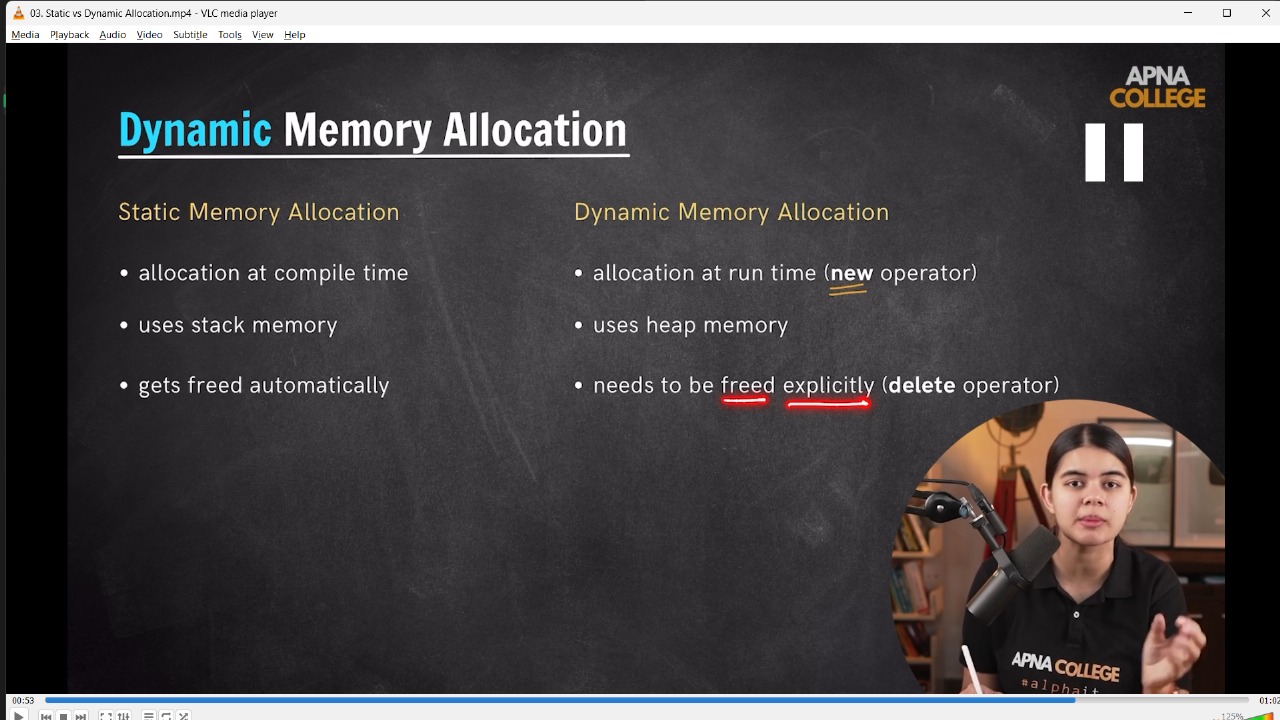




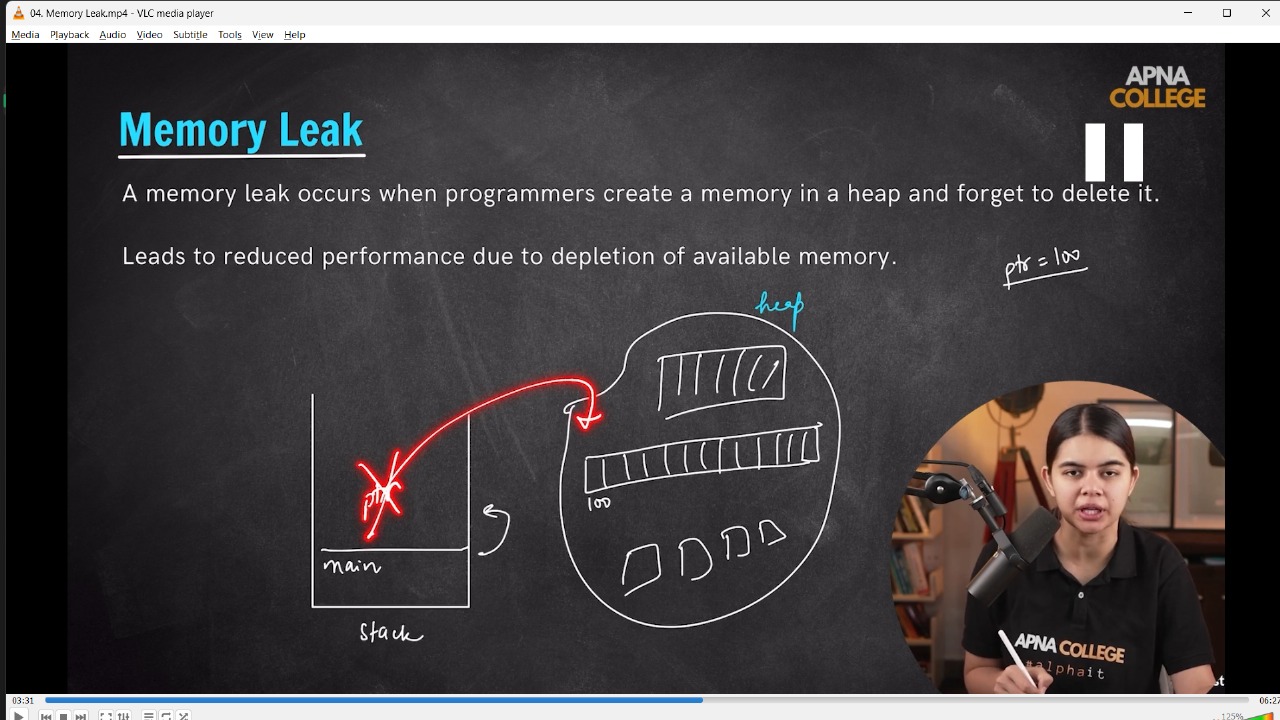


Difference B/w DMA & SMA -



**Memory Leak –**



// int main()

// {

//     /\*

//     // int arr[100] = {1,2,3,4,5};

//     int size;

//     cin >> size;//10

//     int \*arr = new int[size]; // so by using new keyword we can create an arrayu dynamically in memory

//     arr[0] = \*(arr + 0);

//     arr[1] = \*(arr + 1);

//     arr[2] = \*(arr + 2);

//     int x = 1;

//     for (int i = 0; i < size; i++)

//     {

//         arr[i] = x;

//         cout << arr[i] << " ";

//         x++;

//     }

//     cout << endl;

//     /\*

//     10

//     1 2 3 4 5 6 7 8 9 10

//      \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

/\*📒  In SMA - var,aray memo, fun me create hoti he staack me run hoti he within the fun end ho jaati he, no access in main fun or anywhere.

    but in DMA - Memory HEap me create hoti he and throughout the code can be accessible from anywhere

    2)  SO, here using new keyword. it's heap memory, but it'll also appearing every time still when no one sui g it. so as a

    program mandatory to delete the memory too.

    3) Hmne new keyword use kr ke pointer array or var bna liya lekin after using it in the main fun it must to delete again in the respectivve fun as the memory created in heap and will be existing throughout the code untill delete it - then bhale hi use na ho rhi ho

     \*/

// void funcInt()

// {

//     int \*ptr = new int;

//     \*ptr = 5;

//     cout<<\*ptr;

//     delete ptr;

// }

// void funcArray()

// {

//     // int arr[100] = {1,2,3,4};

//     int size;

//     cin>>size;

//     int \*arr = new int[size];// Created Dynamicallt Alllocated Memory

//     int x = 1;

//     for (int i = 0; i < size; i++)

//     {

//         arr[i] = x;

//         cout << arr[i] << " ";

//         x++;

//     }

//     cout<<endl;

//     delete[] arr;// Freeing up the sapce

// }

// int main()

// {

//     funcArray();

//     /\*

//     10

//     1 2 3 4 5 6 7 8 9 10

//      \*/

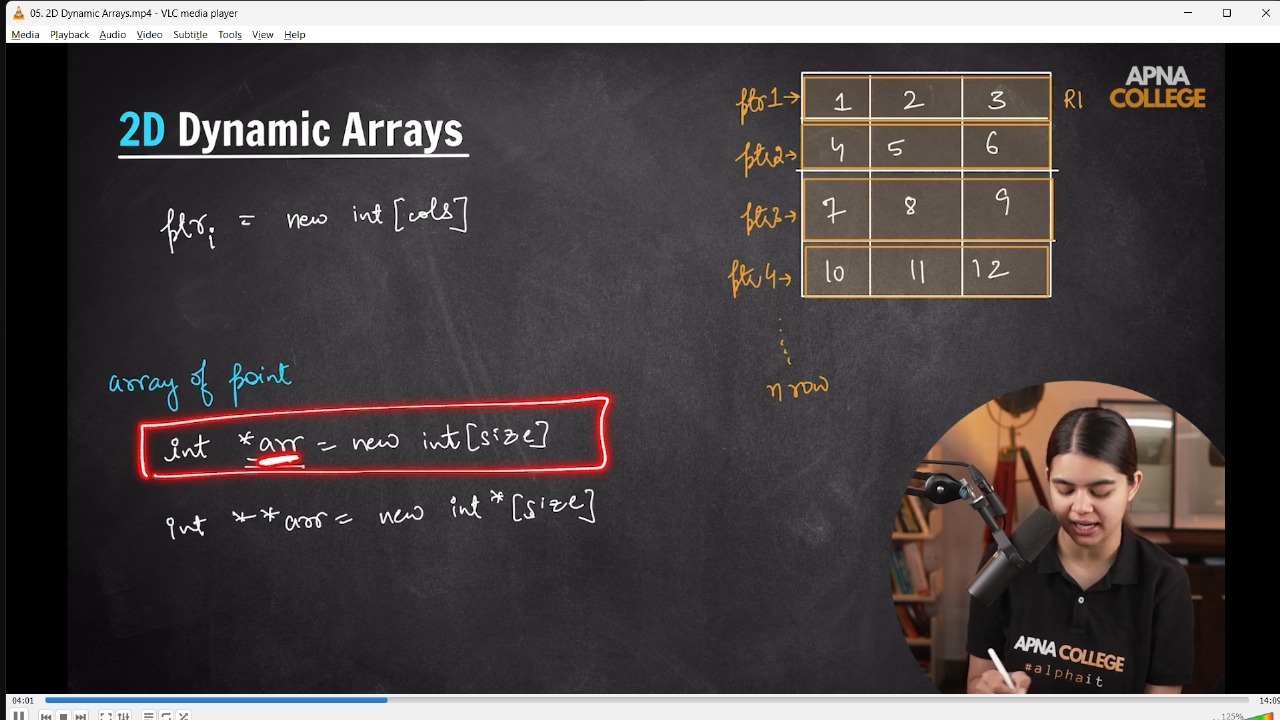
//     funcInt();

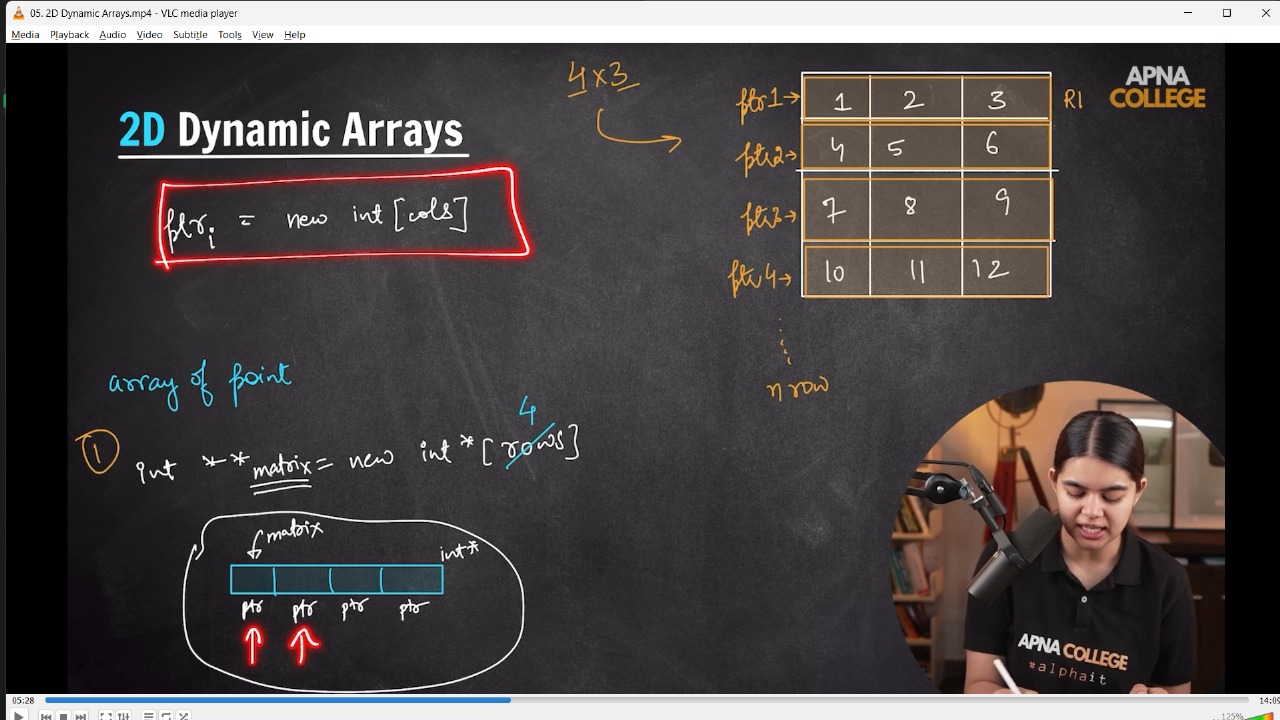
//     // 5

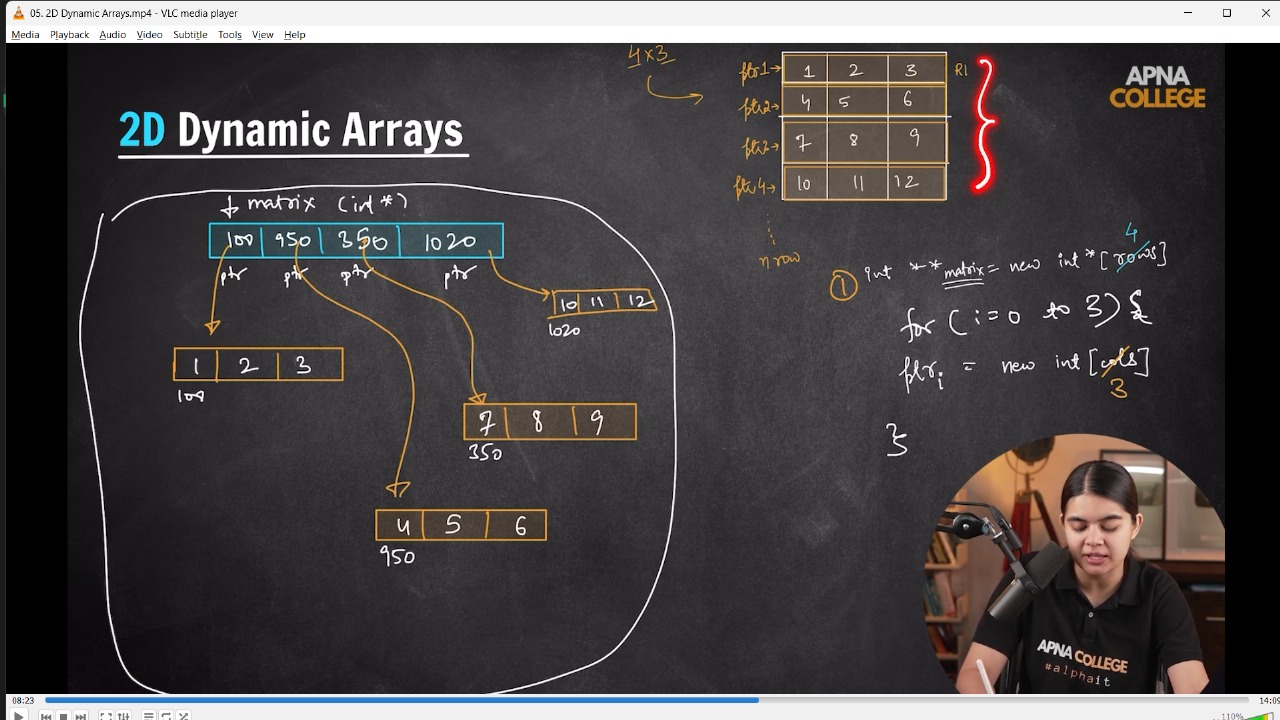
// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//2) DMA in 2D Dynamic Arrays -**







int main()

{

    int n, m;

    cout << "No. of rows & columns are - " << endl;

    cin >> n >> m;

    int \*\*matrix = new int \*[n]; // Correct syntax for 2D dynamic array

    for (int i = 0; i < n; i++)

    {

        matrix[i] = new int[m];

    }

    // Data store

    int x = 1;

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < m; j++)

        {

            matrix[i][j] = x++;

            cout << matrix[i][j] << " ";

        }

        cout << endl;

    }

    // Free the dynamically allocated memory

    for (int i = 0; i < n; i++)

    {

        delete[] matrix[i];

    }

    delete[] matrix;

    return 0;

    /\*

    No. of rows & columns are -

    3

    3

    1 2 3

    4 5 6

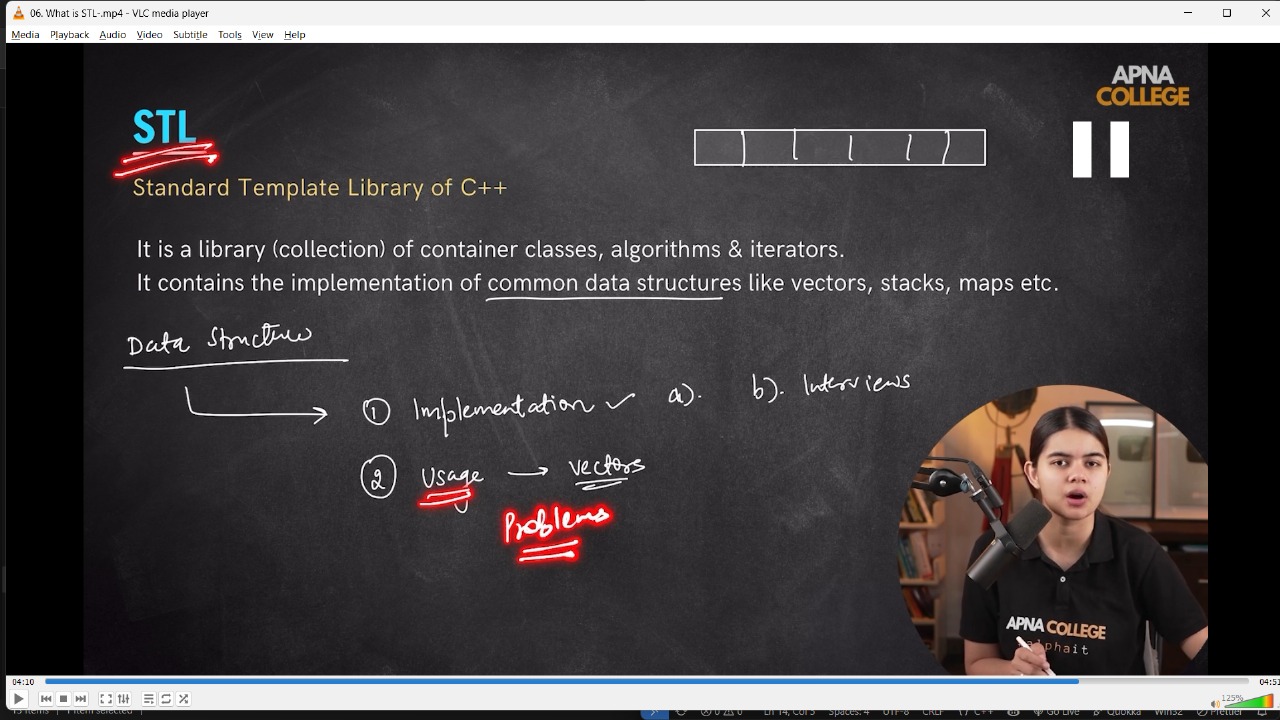
    7 8 9

     \*/

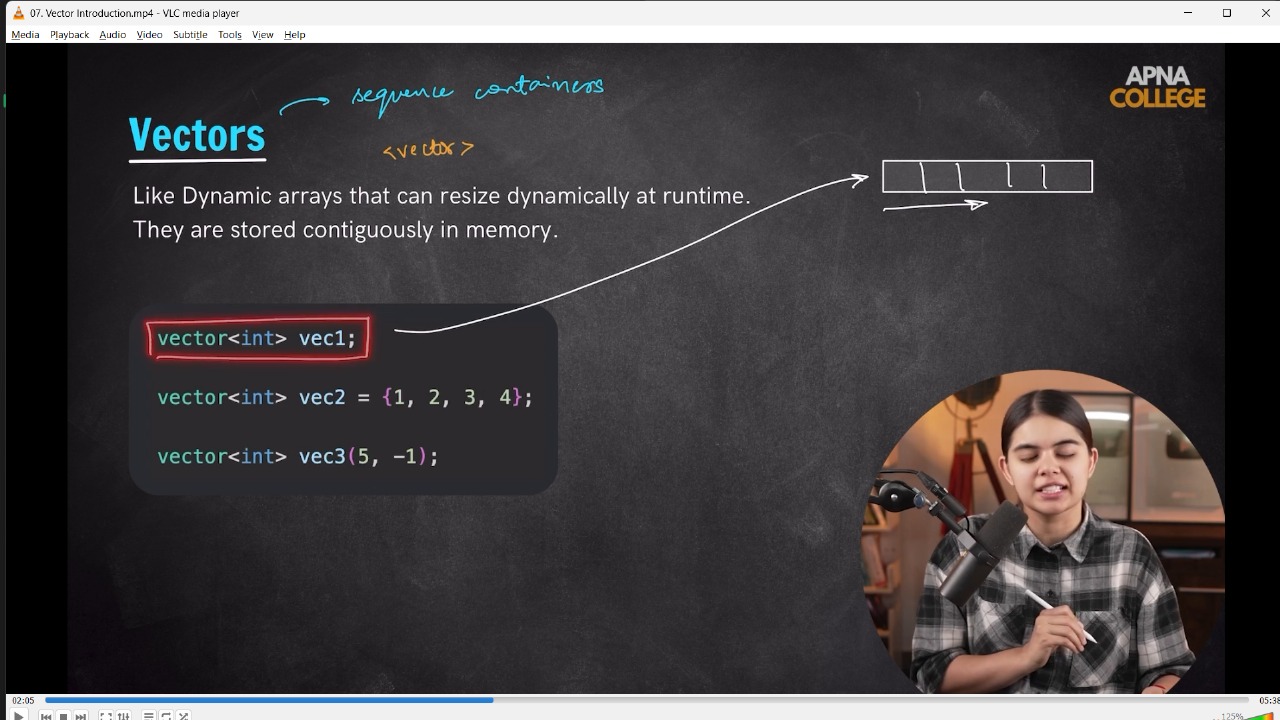
}

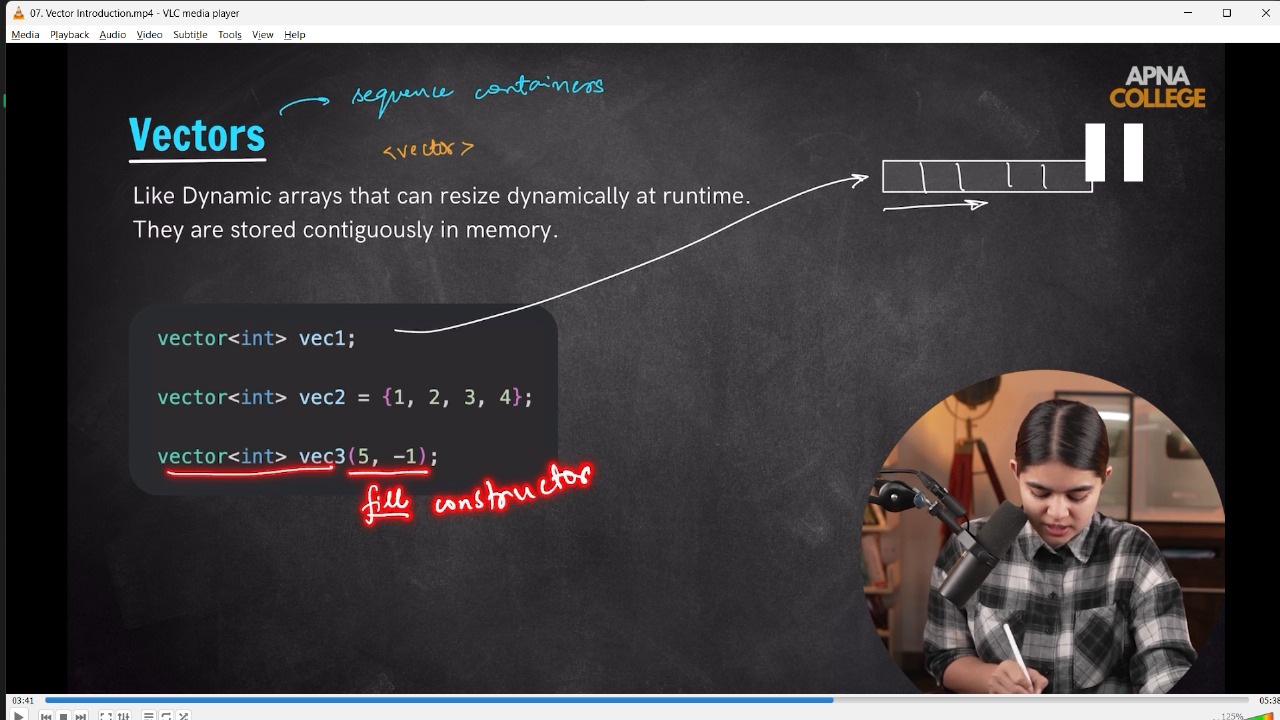
// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

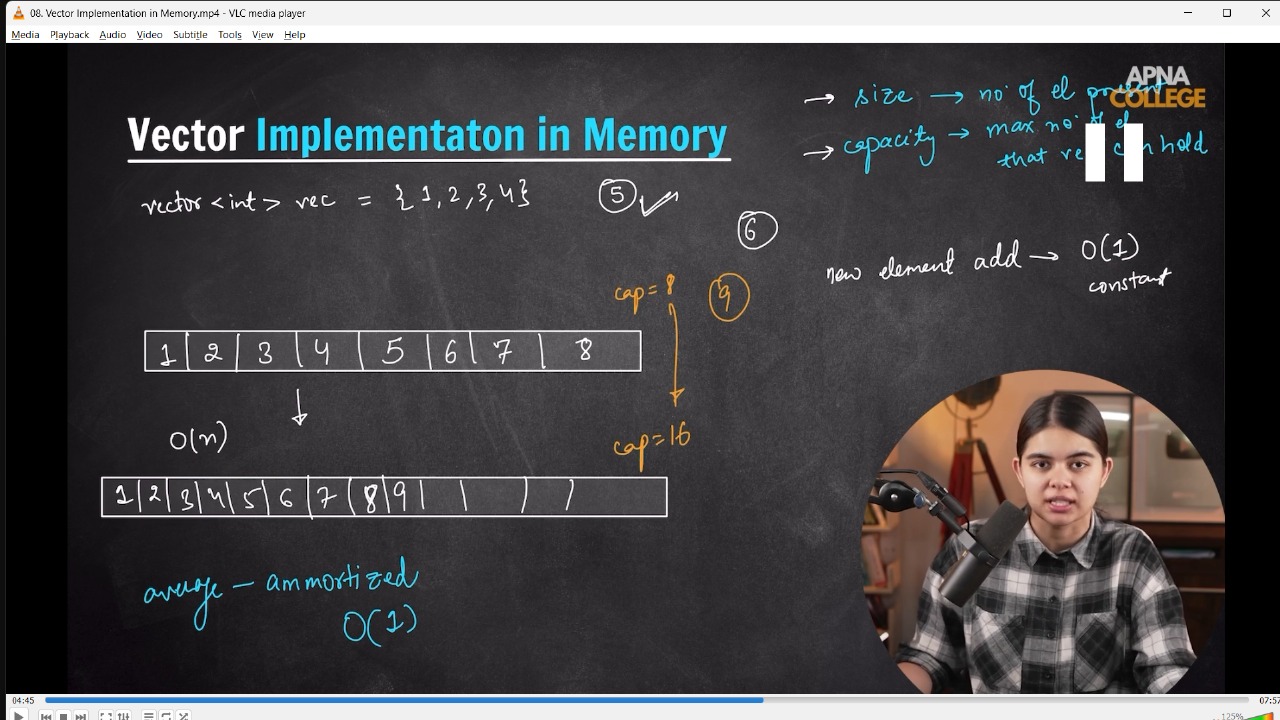
**//3)STL –**

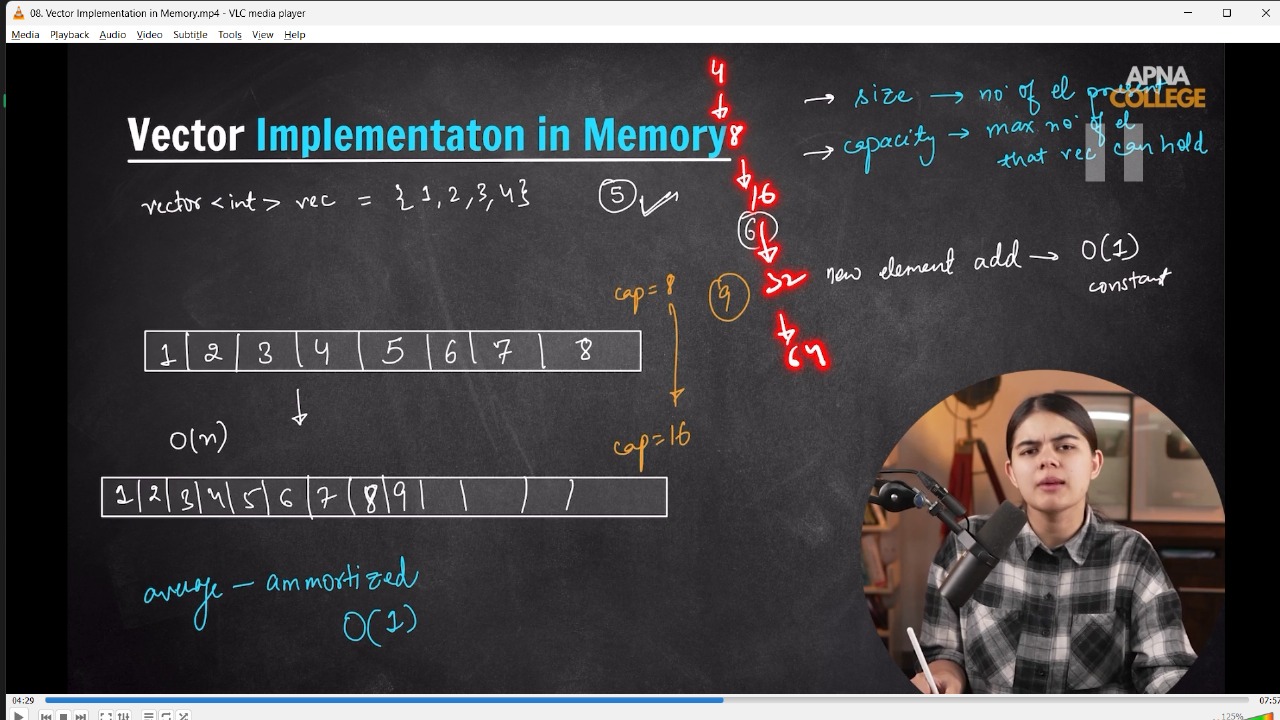


**//3.1) vectors -**









// int main()

// {

// Vector Initialization methods -

//     vector<int> vec1;

//     cout << vec1.size() << endl; // 0

//     vector<int> vec2 = {2, 4, 6, 8};

//     cout << vec2.size() << endl; // 4

//     vector<int> vec3(10, -1);

//     for (int i = 0; i < vec3.size(); i++)

//     {

//         cout << vec3[i] << " ";

//     }

//     cout << endl;

//     cout << vec3.size() << endl; //-1 -1 -1 -1 -1 -1 -1 -1 -1 -1

//     cout << endl;

//     /\*

//     📒 Vector Implementation in Memory - In vector it also deals with derference [ointer cpncepts woithin the memory

//     2 - Vector capacity becomes double of itslef on adding new element

//      \*/

//     vector<int> vec = {1, 2, 3, 4};

//     cout << "Size is - " << vec.size() << endl;         // Size is - 4

//     cout << "Capacity is - " << vec.capacity() << endl; // Capacity is - 4

//     cout << endl;

//     vec.push\_back(5);

//     cout << "Size is - " << vec.size() << endl;         // Size is - 5

//     cout << "Capacity is - " << vec.capacity() << endl; // Capacity is - 8

//     vec.pop\_back();

//     for (auto i: vec)

//     {

//         cout<<vec[i]<<" ";

//     }

//     cout<<endl;

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 3.2) Pair\_Sum - FInd if any pair in Sorted Array has target sum**

**// 3.2.1) Brute Forece Approach - using Nested Loops. TC - O(n^2)**

// int main()

// {

//     int n, target;

//     cout << "vector size - " << endl;

//     cin >> n;

//     vector<int> arr(n);

//     cout << "Cvecrtor Eleemnts are - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, inserted elements of vector are - " << endl;

//     for (auto i : arr)

//     {

//         cout << i << " ";

//     }

//     cout << endl;

//     cout << "target element is - " << endl;

//     cin >> target;

//     for (int i = 0; i < n - 1; i++)

//     {

//         for (int j = i + 1; j < n; j++)

//         {

//             if (arr[i] + arr[j] == target)

//             {

//                 cout << arr[i] << " " << arr[j] << endl;

//                 cout << "Indices: " << i << " " << j << endl;

//             }

//         }

//     }

//     return 0;

//     /\*

//     PS E:\51 LPA\_Shagun\\_Level-Up-Data-Structures> cd "e:\51 LPA\_Shagun\\_Level-Up-Data-Structures\6\_Vectors\" ; if ($?) { g++ 2\_STL.cpp -o 2\_STL } ; if ($?) { .\2\_STL }

//     vector size -

//     5

//     Cvecrtor Eleemnts are -

//     10 12 8 2 5

//     So, inserted elements of vector are -

//     10 12 8 2 5

//     target element is -

//     18

//     10 8

//     Indices: 0 2

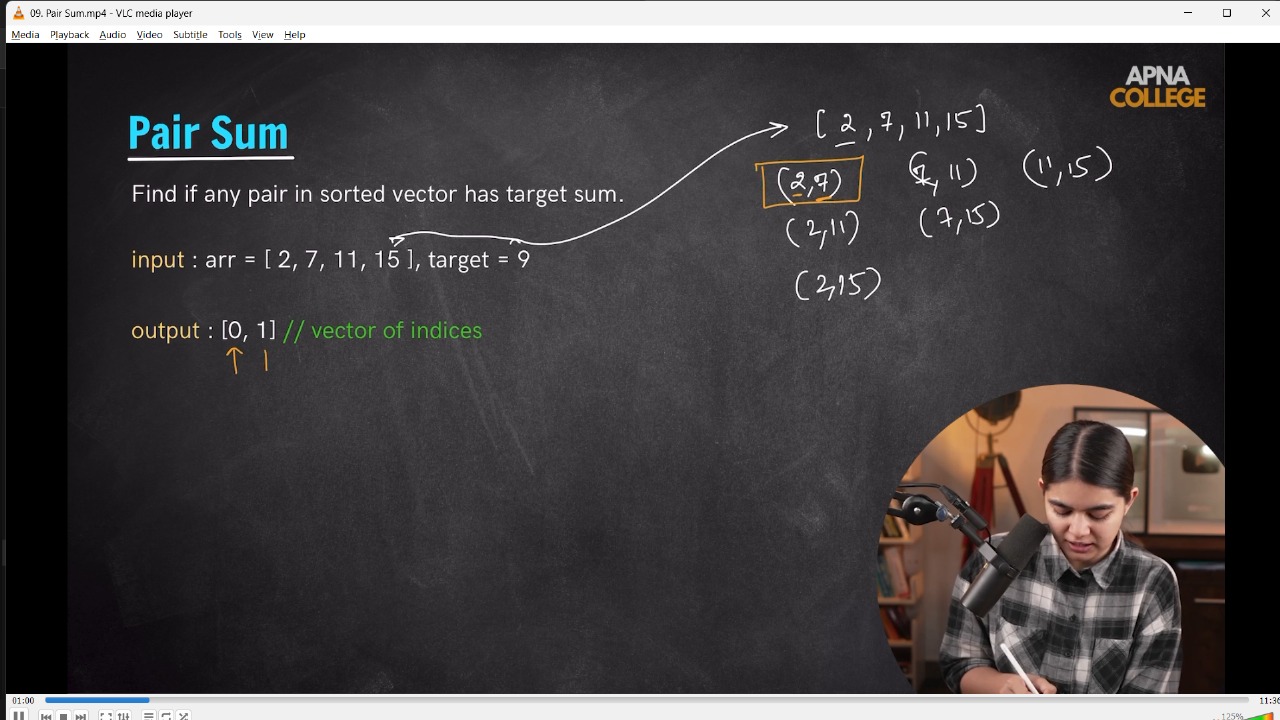
//     TC- O(N^2)

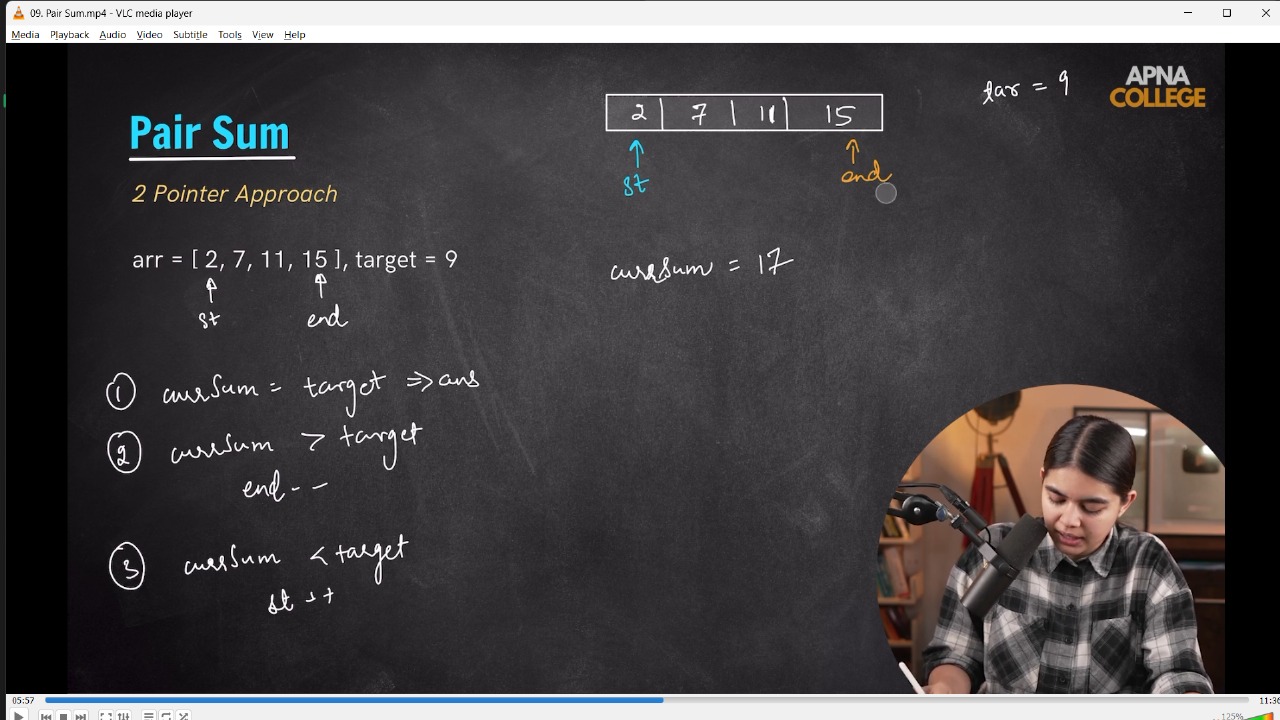
//      \*/

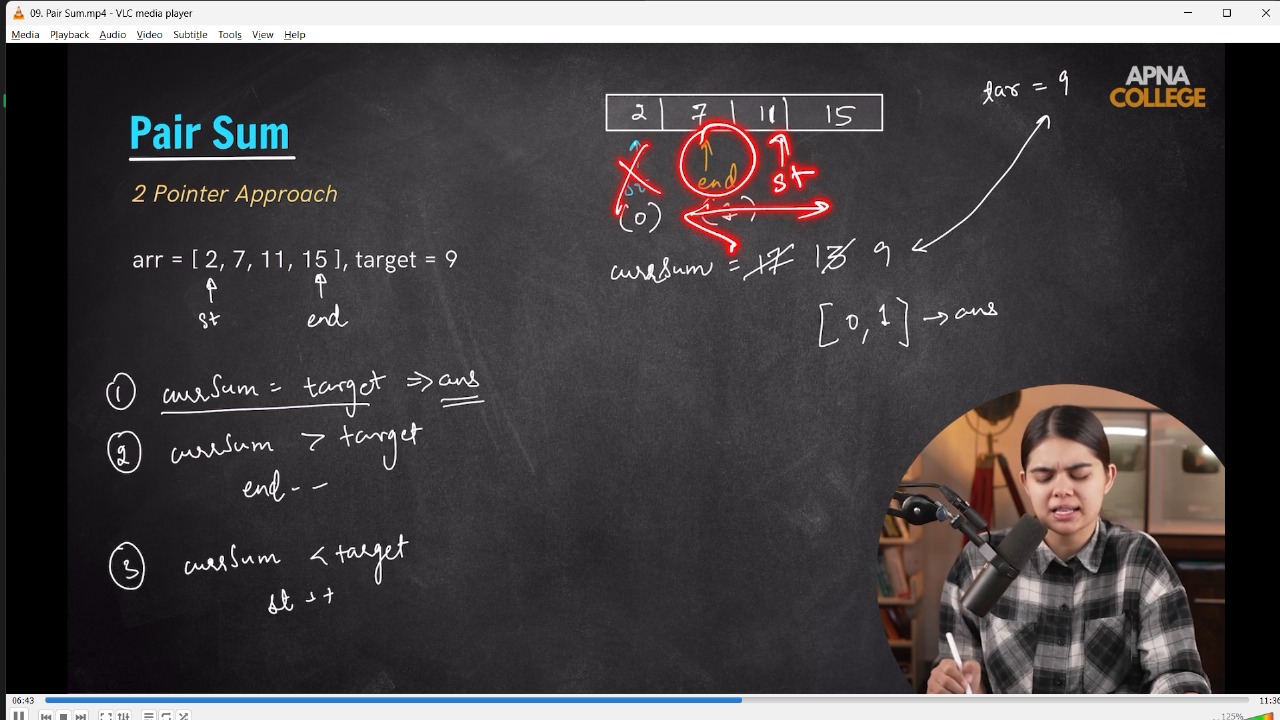
// }

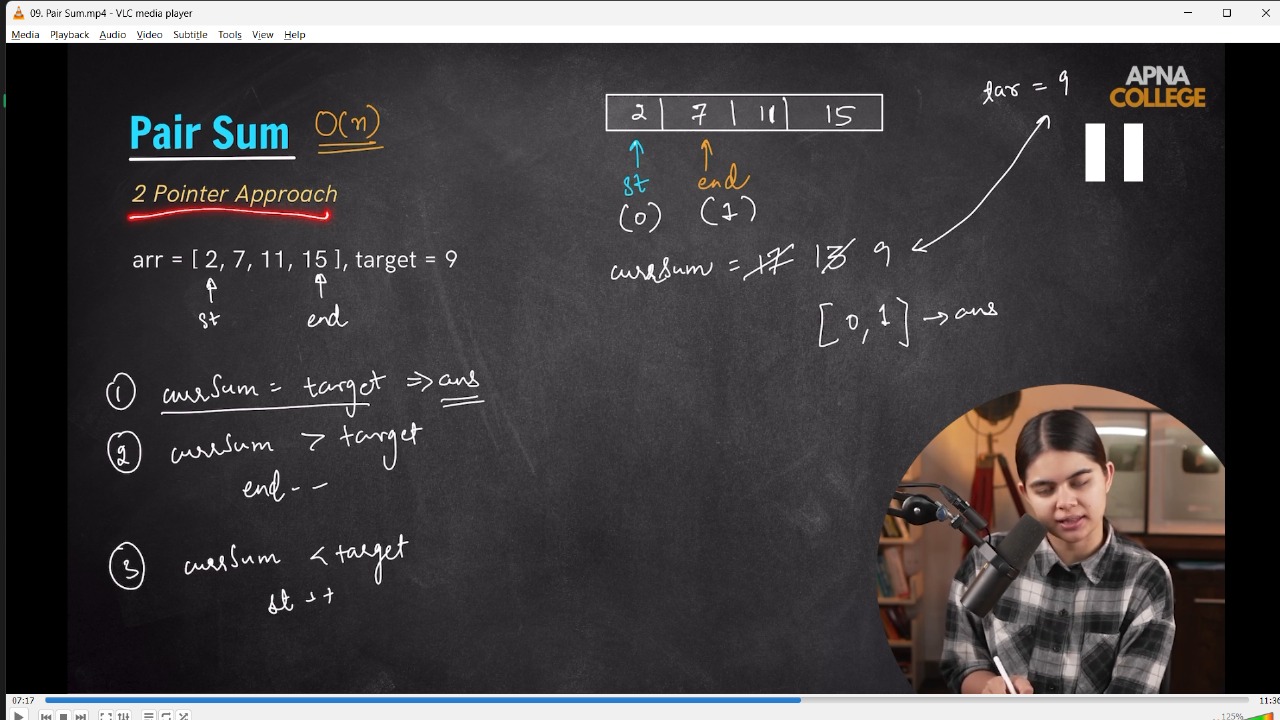
// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//3.2.2) Uisng Linear Approach - 2 Pointer Approach. TC - O(n)**









// vector<int> pairSum(vector<int> vec, int target)

// {

//     int st = 0, end = vec.size() - 1;

//     int currSum = 0;

//     vector<int> Ans;

//     while (st < end)

//     {

//         currSum = vec[st] + vec[end];

//         if (currSum == target)

//         {

//             Ans.push\_back(st);

//             Ans.push\_back(end);

//             return Ans;

//         }

//         else if (currSum > target)

//         {

//             end--;

//         }

//         else

//         {

//             st++;

//         }

//     }

//     return Ans;

// }

// int main()

// {

//     int n;

//     cout << "What is the vector Size here -" << endl;

//     cin >> n;

//     vector<int> vec(n);

//     cout << "What are the Vector Elements - " << endl;

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

//     {

//         cin >> vec[i];

//     }

//     cout << "Hence, the inserted vector is - " << endl;

//     for (auto i : vec)

//     {

//         cout << i << " ";

//     }

//     cout << endl;

//     int target;

//     cout << "Mention the value you want to target - " << endl;

//     cin >> target;

//     vector<int> Ans = pairSum(vec, target);

//     cout << "The indices of the elements that sum to target are: ";

//     for (auto idx : Ans)

//     {

//         cout << idx << " ";

//     }

//     cout << endl;

// /\*

// What is the vector Size here -

// 4

// What are the Vector Elements -

// 2 7 11 15

// Hence, the inserted vector is -

// 2 7 11 15

// Mention the value you want to target -

// 9

// The indices of the elements that sum to target are: 0 1

// ----------------------

// What is the vector Size here -

// 4

// What are the Vector Elements -

// 2 7 11 15

// Hence, the inserted vector is -

// 2 7 11 15

// Mention the value you want to target -

// 17

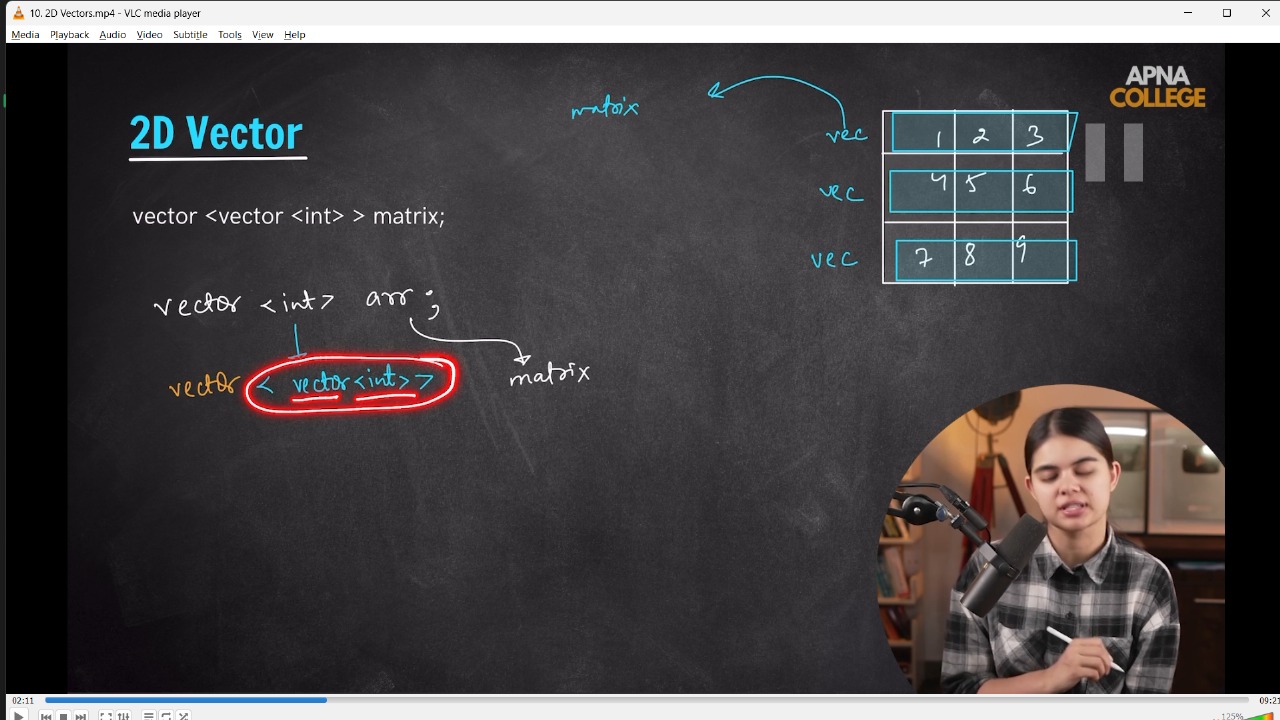
// The indices of the elements that sum to target are: 0 3

//  \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 4) 2-D Vector -**



//  int main()

//  {

//      vector<vector<int>> Matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

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

//     {

//         for (int j = 0; j < Matrix[i].size(); j++)

//         {

//             cout << Matrix[i][j] << " ";

//         }

//         cout << endl;

//     }

//     /\*

//     1 2 3

//     4 5 6

//     7 8 9

//      \*/

// // -----------------

//     // In 2D Vector no issue of assigning row and colm. It can be done in any way easily

//     vector<vector<int>> Matrix1 = {{1, 2, 3}, {4, 5}, {6}};

//     for (int i = 0; i < Matrix1.size(); i++)

//     {

//         for (int j = 0; j < Matrix1[i].size(); j++)

//         {

//             cout << Matrix1[i][j] << " ";

//         }

//         cout << endl;

//     }

// /\*

// 1 2 3

// 4 5

// 6

//  \*/

// // -----------------

//     // Similarly by trying from taking the input from the user -

//     int n, m;

//     cout << "Enter the number of rows & columns respectively: ";

//     cin >> n >> m;

//     vector<vector<int>> M2(n, vector<int>(m)); // Proper 2D vector initialization

//     cout << "Mention the vector elements:" << endl;

//     for (int i = 0; i < n; i++)

//     {

//         for (int j = 0; j < m; j++)

//         {

//             cin >> M2[i][j];

//         }

//     }

//     cout << "The inserted elements of the vector are:" << endl;

//     for (int i = 0; i < n; i++)

//     {

//         for (int j = 0; j < m; j++)

//         {

//             cout << M2[i][j] << " ";

//         }

//         cout << endl;

//     }

// /\*

// Enter the number of rows & columns respectively: 4 4

// Mention the vector elements:

// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

// The inserted elements of the vector are:

// 1 2 3 4

// 5 6 7 8

// 9 10 11 12

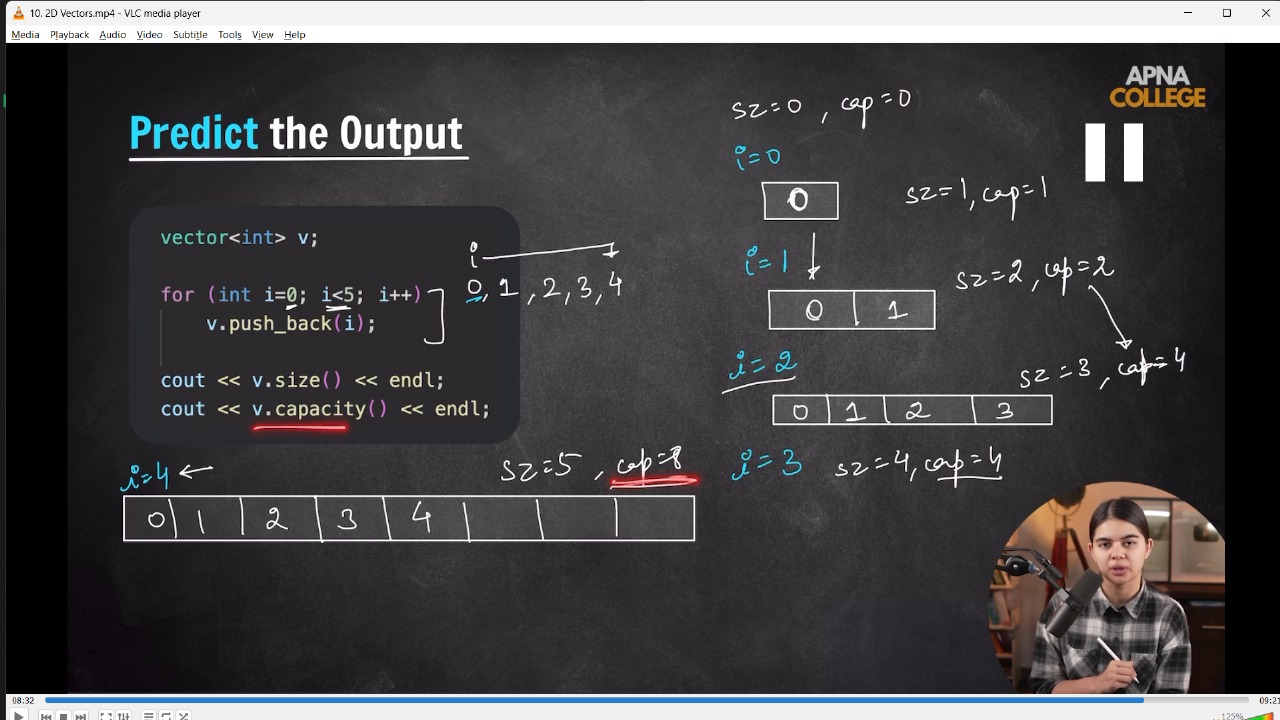
// 13 14 15 16

//  \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// Qun) What will be the o/p -**

 // int main()

// {

//     vector<int> v;

//     for (int i = 0; i < 5; i++)

//     {

//         v.push\_back(i);

//     }

//     cout<<v.size()<<endl;// 5

//     cout<<v.capacity()<<endl;// 8

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_