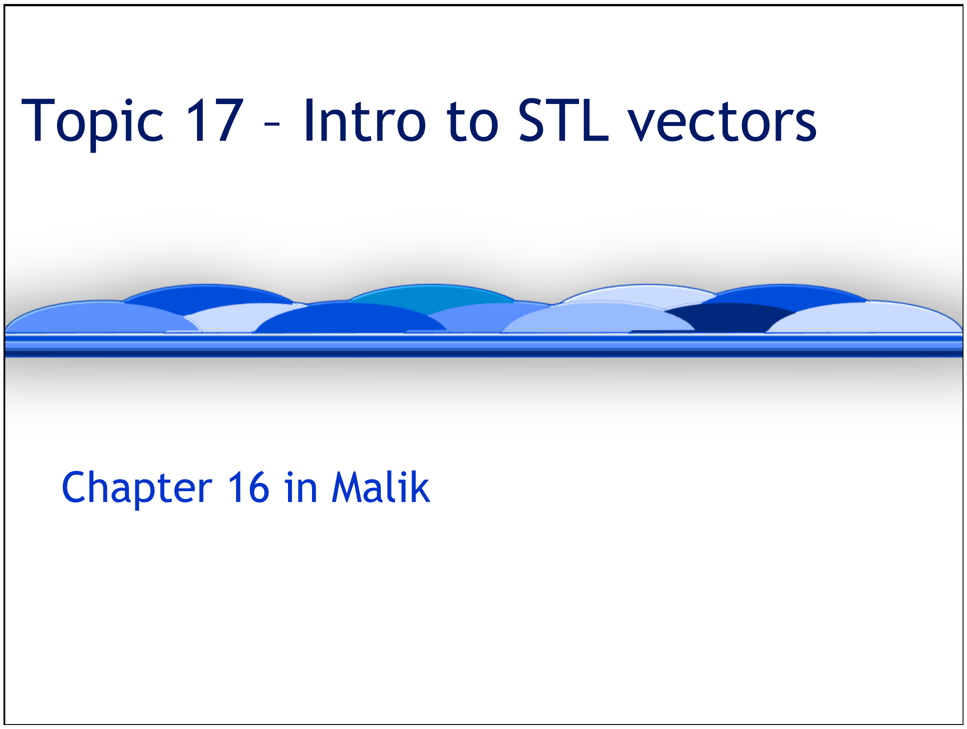
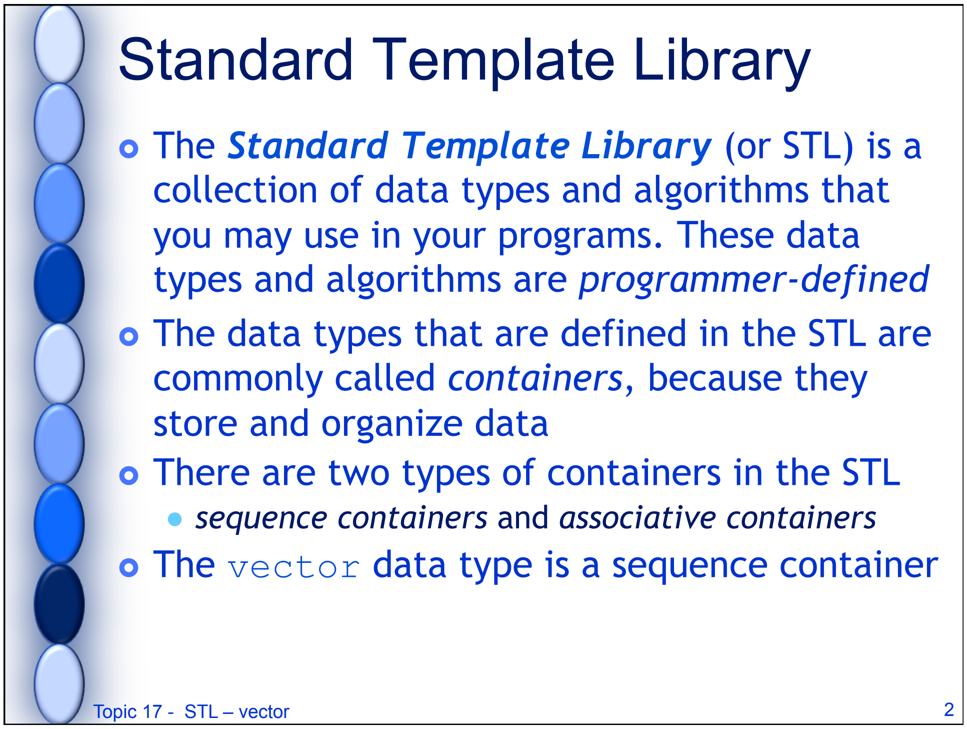


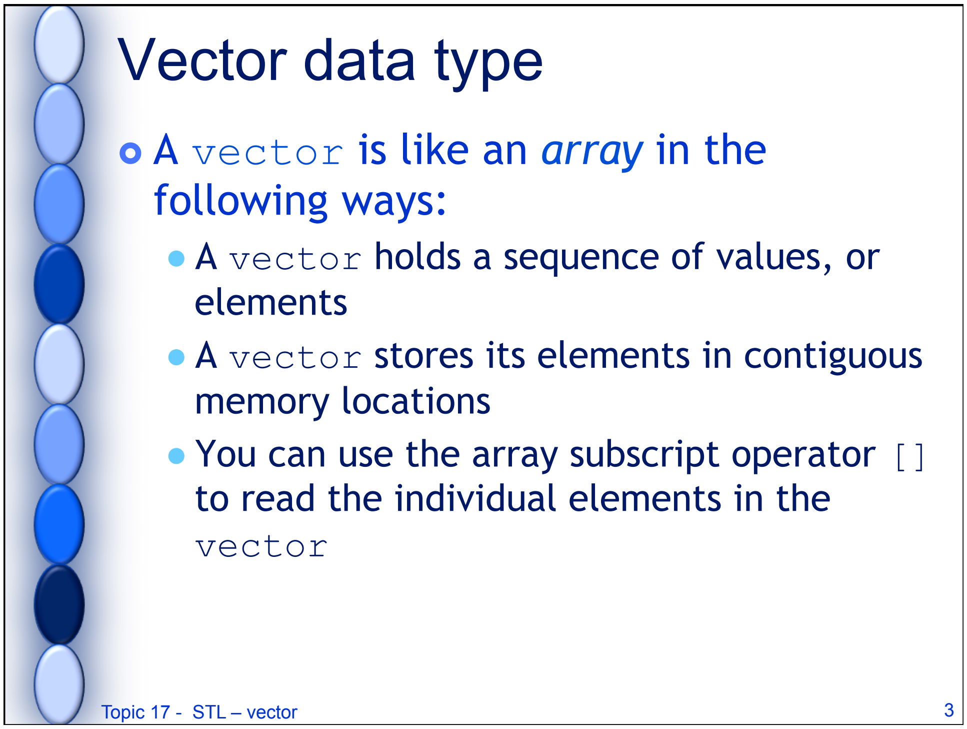
Topic 17 – Intro to STL vectors

Chapter 16 in Malik



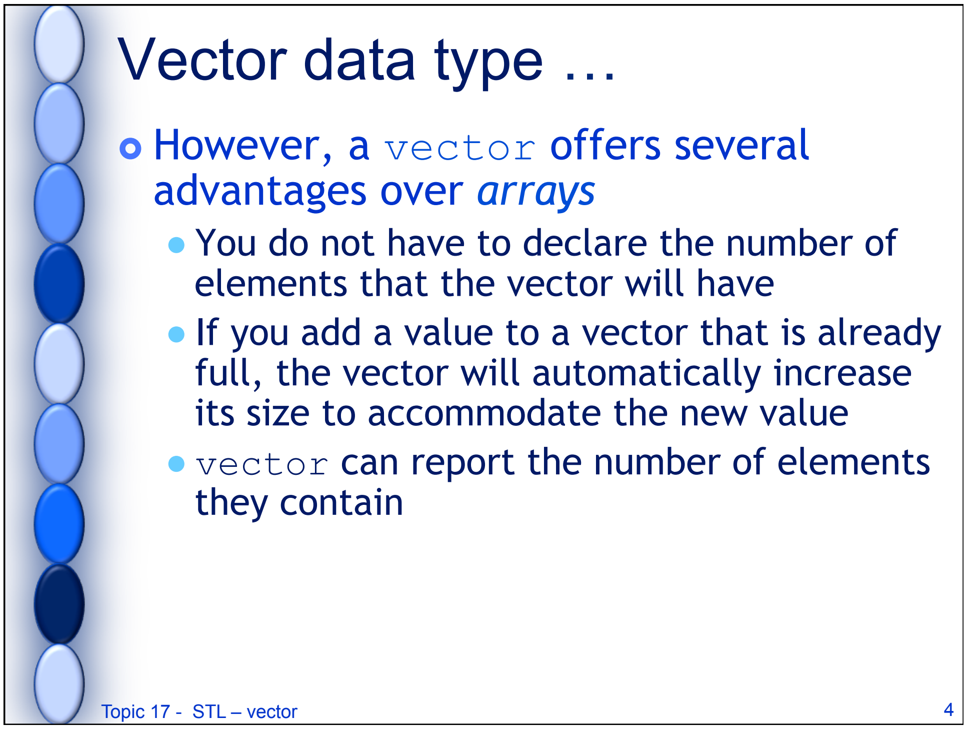
Standard Template Library

- The ***Standard Template Library*** (or STL) is a collection of data types and algorithms that you may use in your programs. These data types and algorithms are *programmer-defined*
- The data types that are defined in the STL are commonly called *containers*, because they store and organize data
- There are two types of containers in the STL
  - *sequence containers* and *associative containers*
- The `vector` data type is a sequence container



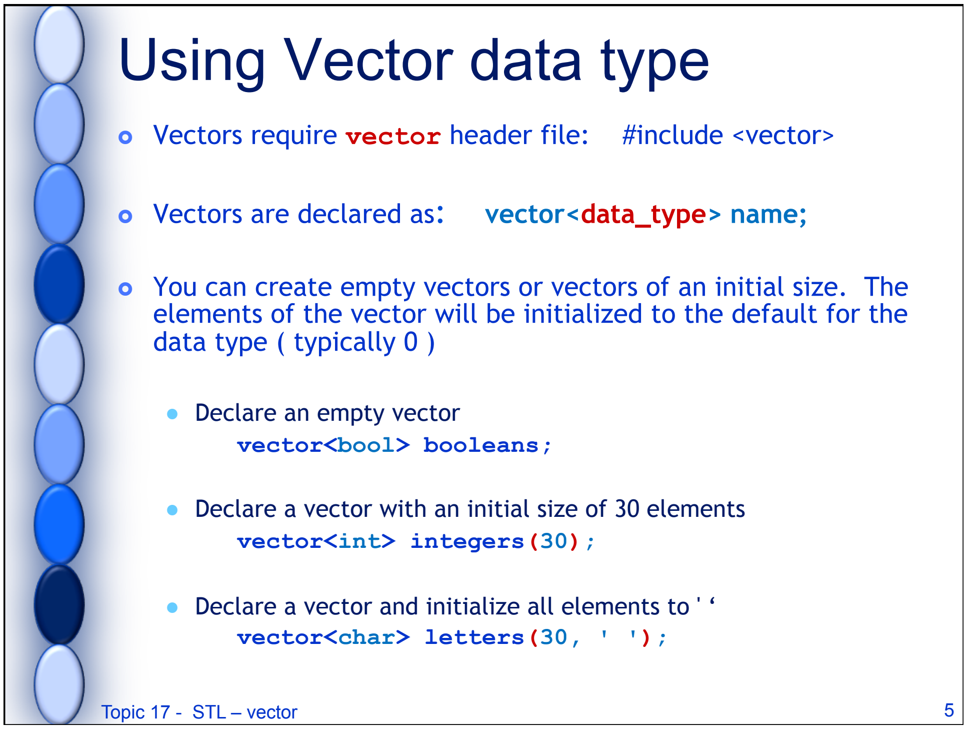
Vector data type

- A `vector` is like an *array* in the following ways:
  - A `vector` holds a sequence of values, or elements
  - A `vector` stores its elements in contiguous memory locations
  - You can use the array subscript operator `[]` to read the individual elements in the `vector`



Vector data type …

- However, a `vector` offers several advantages over *arrays*
  - You do not have to declare the number of elements that the vector will have
  - If you add a value to a vector that is already full, the vector will automatically increase its size to accommodate the new value
  - `vector` can report the number of elements they contain



Using Vector data type

- Vectors require **vector** header file: #include <vector>

- Vectors are declared as: vector<data_type> name;

- You can create empty vectors or vectors of an initial size. The elements of the vector will be initialized to the default for the data type (typically 0)

  - Declare an empty vector

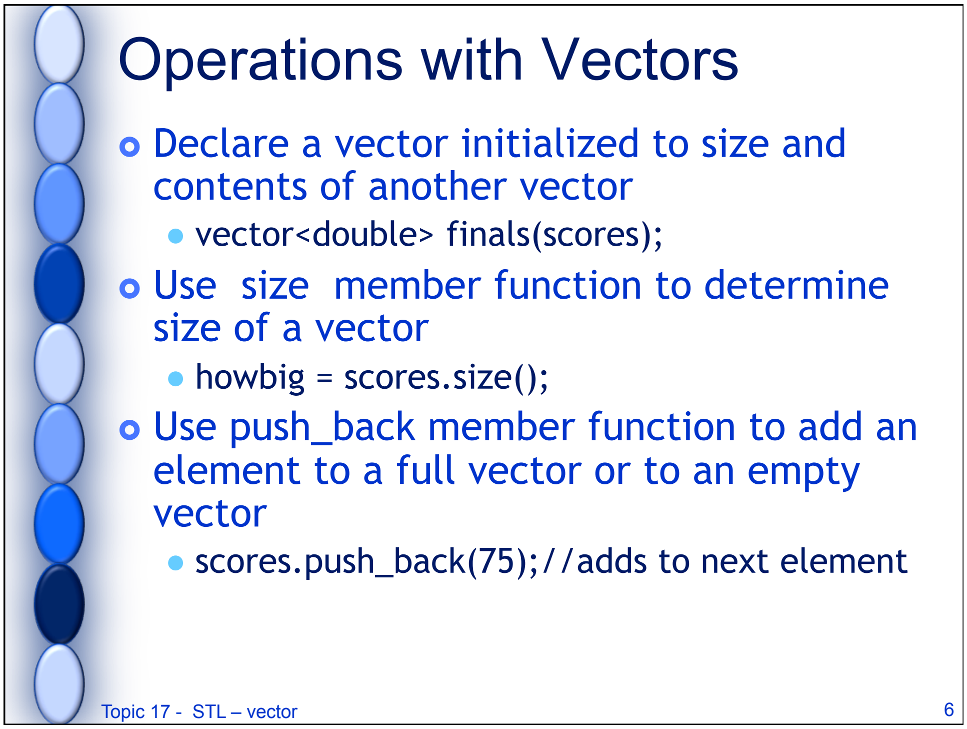
    ```
    vector<bool> booleans;
    ```

  - Declare a vector with an initial size of 30 elements

    ```
    vector<int> integers(30);
    ```

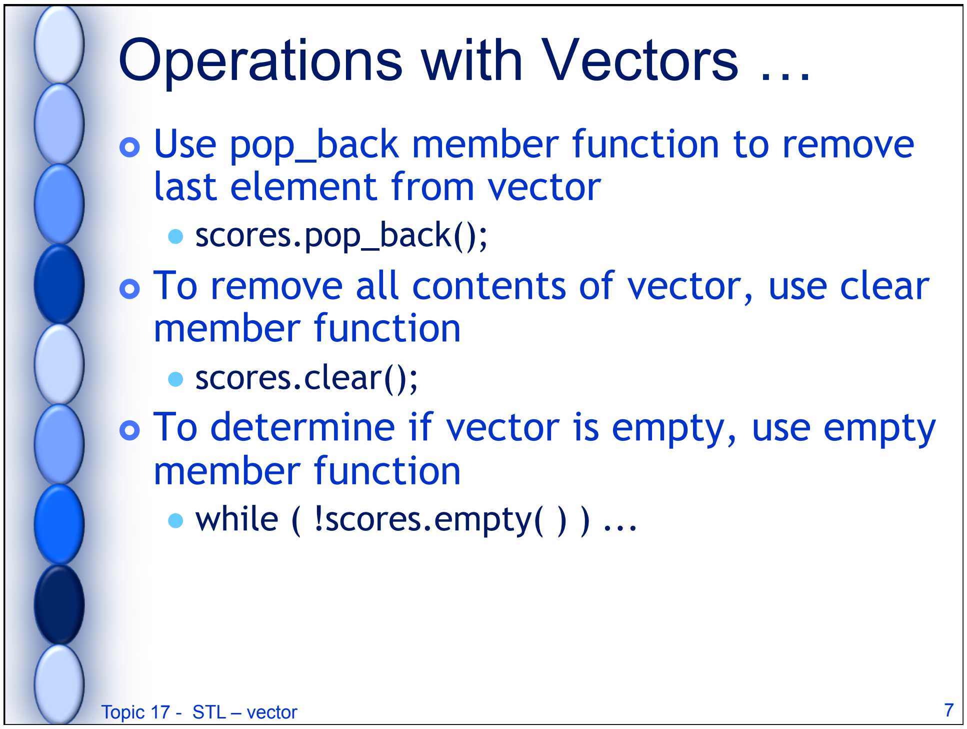
  - Declare a vector and initialize all elements to ' '

    ```
    vector<char> letters(30, ' ');
    ```



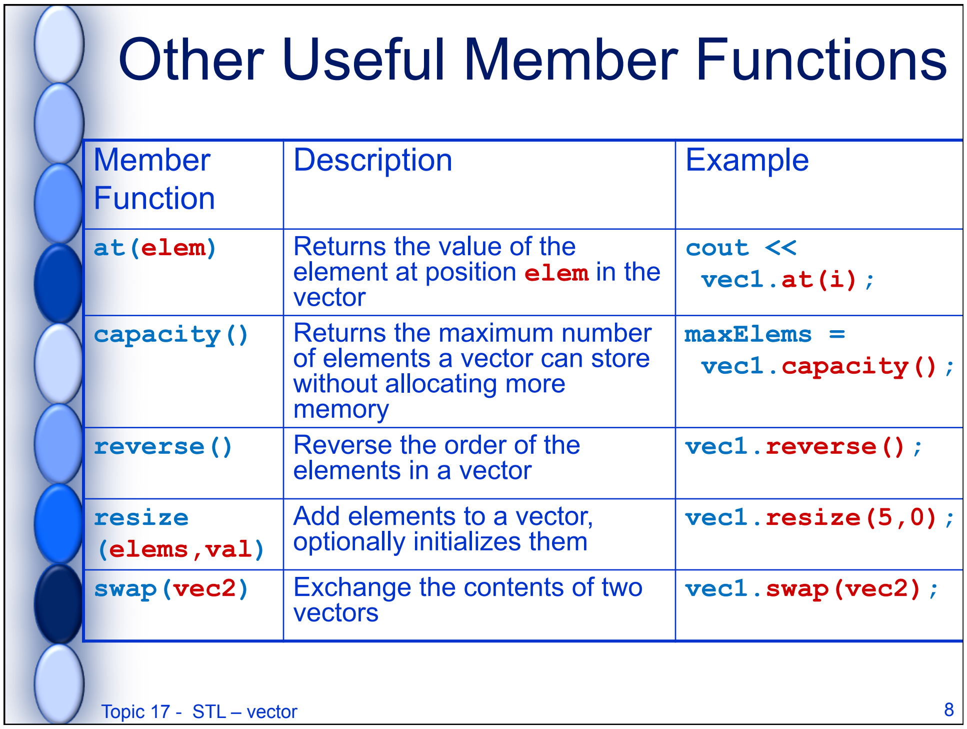
Operations with Vectors

- Declare a vector initialized to size and contents of another vector
  - vector<double> finals(scores);
- Use size member function to determine size of a vector
  - howbig = scores.size();
- Use push_back member function to add an element to a full vector or to an empty vector
  - scores.push_back(75);//adds to next element



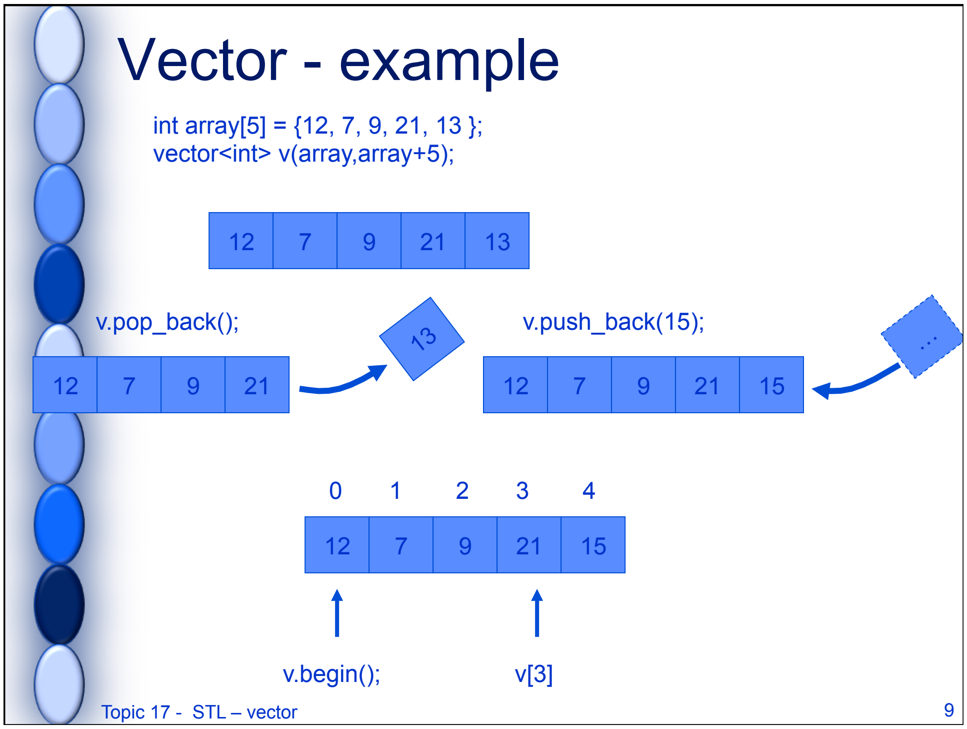
Operations with Vectors …

- Use pop_back member function to remove last element from vector
  - scores.pop_back();
- To remove all contents of vector, use clear member function
  - scores.clear();
- To determine if vector is empty, use empty member function
  - while (!scores.empty( )) …



Other Useful Member Functions

Member Function	Description	Example
`at(elem)`	Returns the value of the element at position **elem** in the vector	`cout << vec1.at(i);`
`capacity()`	Returns the maximum number of elements a vector can store without allocating more memory	`maxElems = vec1.capacity();`
`reverse()`	Reverse the order of the elements in a vector	`vec1.reverse();`
`resize (elems,val)`	Add elements to a vector, optionally initializes them	`vec1.resize(5,0);`
`swap(vec2)`	Exchange the contents of two vectors	`vec1.swap(vec2);`



Vector - example

```
int array[5] = {12, 7, 9, 21, 13 };
vector<int> v(array,array+5);
```

12	7	9	21	13

v.pop_back();

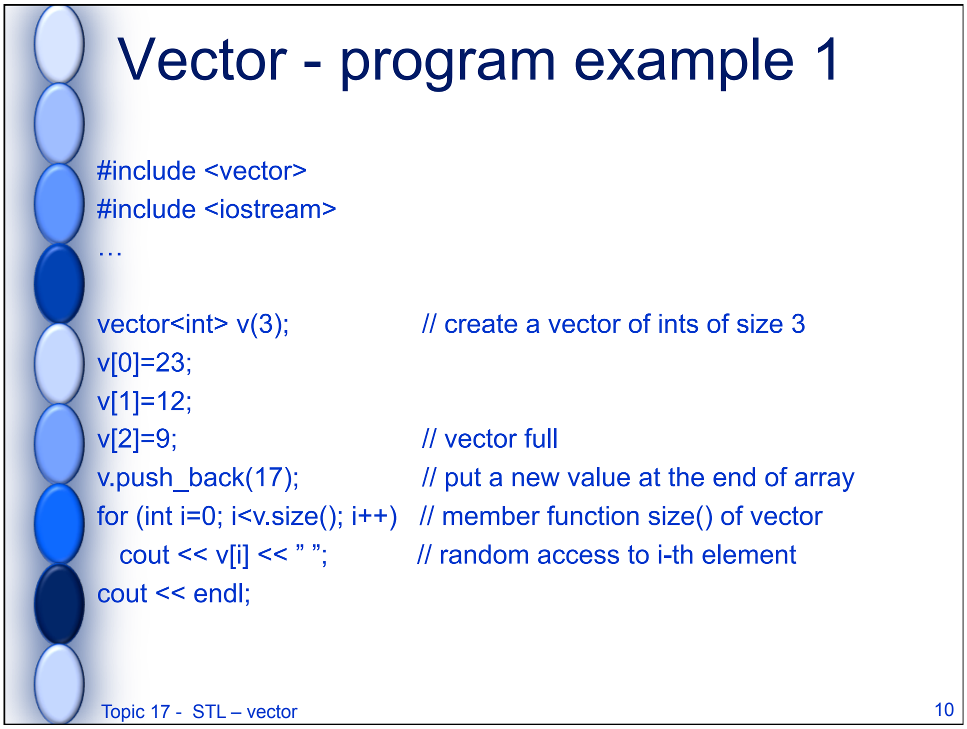
12	7	9	21

v.push_back(15);

12	7	9	21	15

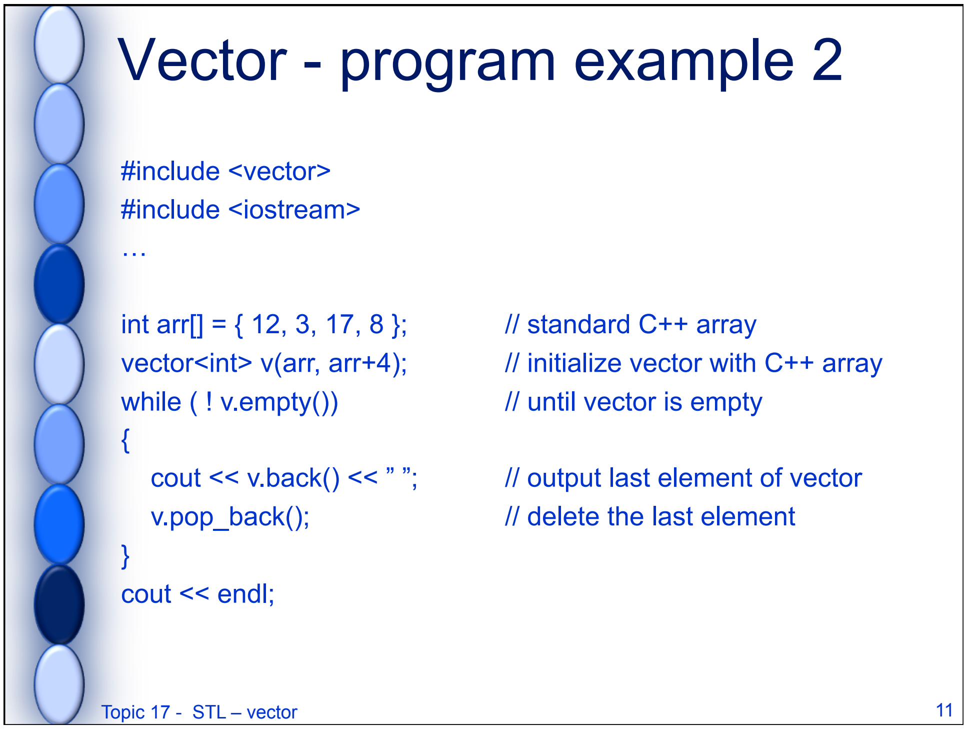
0	1	2	3	4
12	7	9	21	15

v.begin(); v[3]



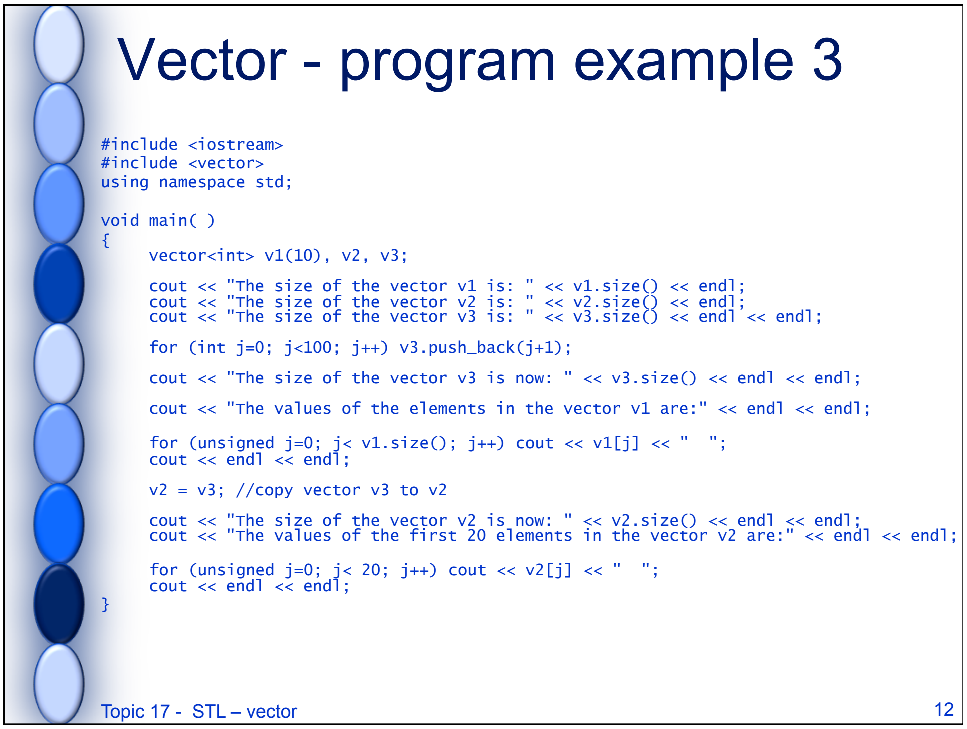
Vector - program example 1

```
#include <vector>
#include <iostream>
…

vector<int> v(3);               // create a vector of ints of size 3
v[0]=23;
v[1]=12;
v[2]=9;                         // vector full
v.push_back(17);                // put a new value at the end of array
for (int i=0; i<v.size(); i++)  // member function size() of vector
   cout << v[i] << ” ”;         // random access to i-th element
cout << endl;
```



Vector - program example 2

```
#include <vector>
#include <iostream>
…

int arr[] = { 12, 3, 17, 8 };          // standard C++ array
vector<int> v(arr, arr+4);             // initialize vector with C++ array
while ( ! v.empty())                   // until vector is empty
{
   cout << v.back() << " ";            // output last element of vector
   v.pop_back();                       // delete the last element
}
cout << endl;
```



Vector - program example 3

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

void main( )
{
     vector<int> v1(10), v2, v3;

     cout << "The size of the vector v1 is: " << v1.size() << endl;
     cout << "The size of the vector v2 is: " << v2.size() << endl;
     cout << "The size of the vector v3 is: " << v3.size() << endl << endl;

     for (int j=0; j<100; j++) v3.push_back(j+1);

     cout << "The size of the vector v3 is now: " << v3.size() << endl << endl;

     cout << "The values of the elements in the vector v1 are:" << endl << endl;

     for (unsigned j=0; j< v1.size(); j++) cout << v1[j] << "  ";
     cout << endl << endl;

     v2 = v3; //copy vector v3 to v2

     cout << "The size of the vector v2 is now: " << v2.size() << endl << endl;
     cout << "The values of the first 20 elements in the vector v2 are:" << endl << endl;

     for (unsigned j=0; j< 20; j++) cout << v2[j] << "  ";
     cout << endl << endl;
}
```