Chapter 6: Vectors

CSE 2010 Week 9

Arrays walked so that Vectors could run

- Arrays are all fine and dandy, but one big limitation to them is that they cannot change in size once they have been defined. There might be times when we want to dynamically allocate memory and change the size of the container. This is where vectors come in.
- **Definition:**
  - Vectors are dynamically allocated arrays provided by the C++ Standard Template Library (STL).
  - They are data structures that store an indexed sequence of elements of the same data type, and can change in size.
- To use vectors, we must
  - `#include <vector>`

Declaring and Initializing Vectors

- `vector<datatype> vectorName;`
    - Creates an empty vector. No mem allocated.

      Example: `vector<int> x;`

Element Value	EMPTY
Index #	

- `vector<datatype> vectorName(n);`
    - Creates a vector of n elements, all with a default value

      Example: `vector<int> y(3);`

Element Value	0	0	0
Index #	0	1	2

- `vector<datatype> vectorName(n, default value);`
    - Creates a vector of n elements, with specified value

      Example: `vector<int> z(4,100);`

Element Value	100	100	100	100
Index #	0	1	2	3

- `vector<datatype> vectorName{element1, element2…element n};`
    - Creates a vector with n elements

      Example: `vector<int> w{3,20,8};`

Element Value	3	20	8
Index #	0	1	2

Vector Functions

- The vector class includes many member functions that allow us to modify or get information about a vector (similar to the string class).
- Syntax for using vector member functions:

```
vectorName.functionName(); //must invoke with an existing vector
```

- Adding/deleting elements:
  - **`push_back(n) : appends the value n to the end of a vector`**
    ```
    vector<int> x; //x is empty
    x.push_back(4); // x = 4
    x.push_back(6); // x = 4, 6
    ```
  - **`pop_back(); deletes the last element in a vector`**
    ```
    vector<int> y{5,6,7}; // y = 5,6,7
    y.pop_back(); // y = 5,6
    ```

- Vectors are considered last in, first out data structures. So when you add or delete elements, it does so through the end of the vector.

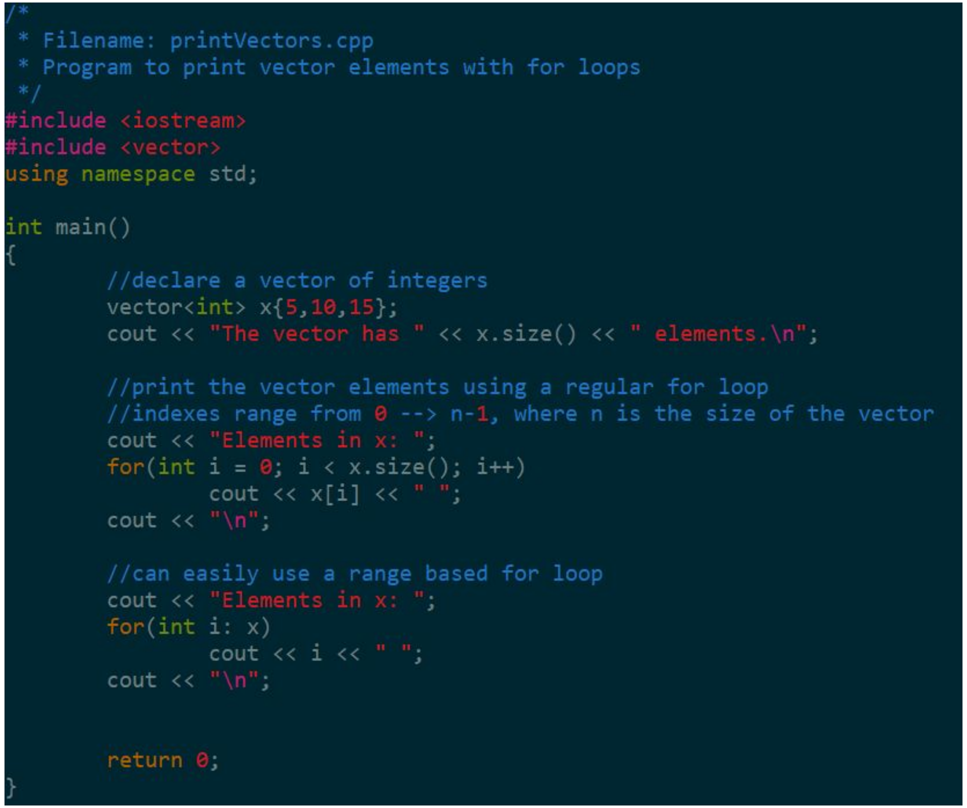
Vector Functions

- Other helpful functions
- `size()`: returns the size of a vector
    - Because vectors are dynamically allocated and their size can change, you can add or delete as many elements as you want. To find out the actual size of a vector you can use .size() (similar to .length() for string)
- `clear()`: clears the vector of all elements, setting its size to 0
- `empty()`: // returns true if the vector has a size of 0, returns false otherwise

Visit : http://www.cplusplus.com/reference/vector/vector/

For a full list of vector member functions and more information on vectors

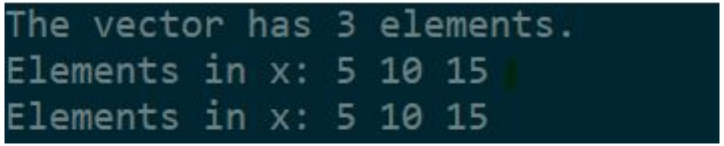
Printing Vector Elements (printVector.cpp)



```
/*
 * Filename: printVectors.cpp
 * Program to print vector elements with for loops
 */
#include <iostream>
#include <vector>
using namespace std;

int main()
{
        //declare a vector of integers
        vector<int> x{5,10,15};
        cout << "The vector has " << x.size() << " elements.\n";

        //print the vector elements using a regular for loop
        //indexes range from 0 --> n-1, where n is the size of the vector
        cout << "Elements in x: ";
        for(int i = 0; i < x.size(); i++)
                cout << x[i] << " ";
        cout << "\n";

        //can easily use a range based for loop
        cout << "Elements in x: ";
        for(int i: x)
                cout << i << " ";
        cout << "\n";


        return 0;
}
```

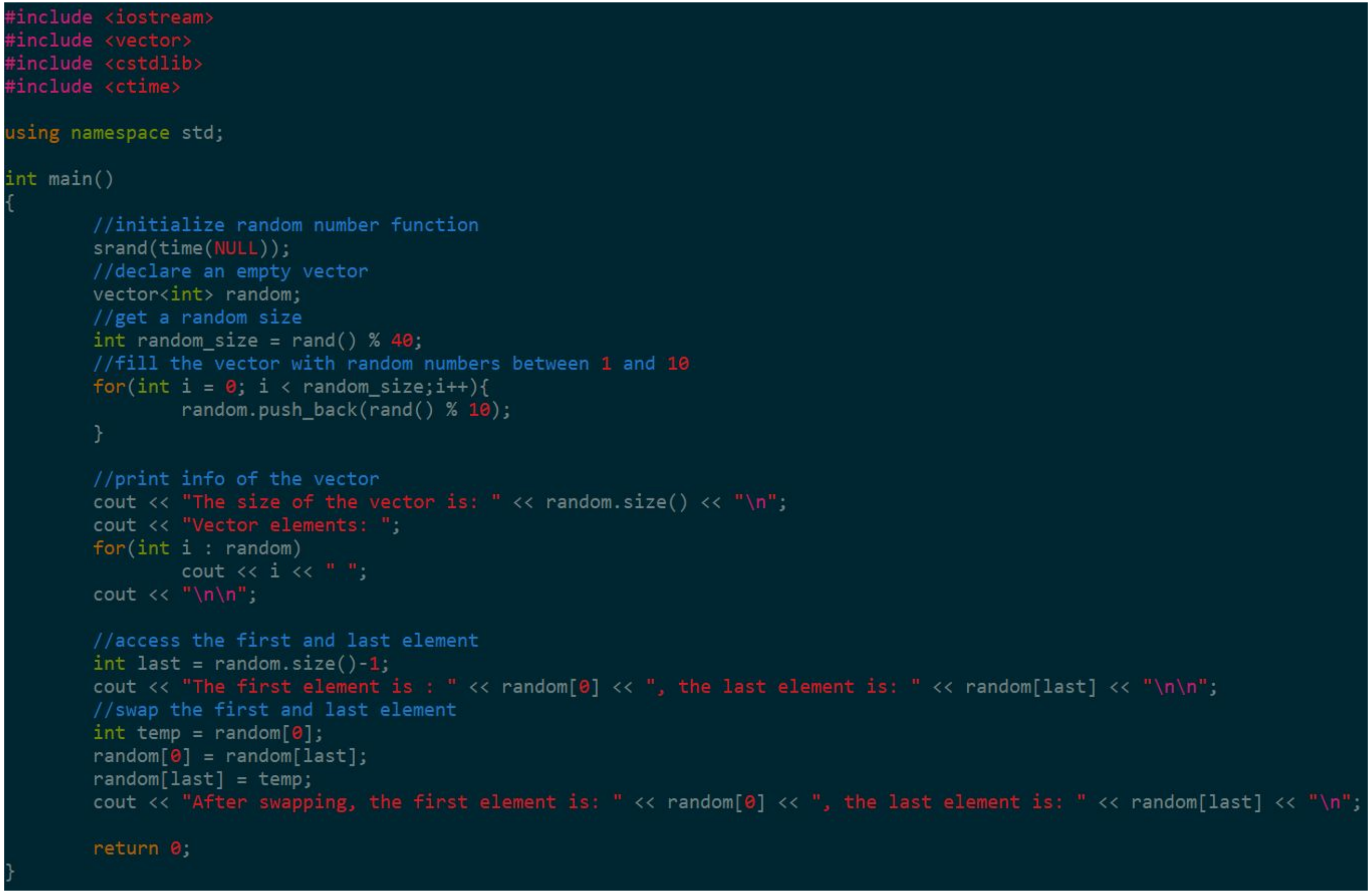
- Like arrays, we can access specific vector elements with the [ ] operator by providing an index #.
- We could use this method or a range based for loop to print vector elements.

Output:



```
The vector has 3 elements.
Elements in x: 5 10 15
Elements in x: 5 10 15
```

Modifying Vector Elements (changeVector.cpp)



```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>

using namespace std;

int main()
{
	//initialize random number function
	srand(time(NULL));
	//declare an empty vector
	vector<int> random;
	//get a random size
	int random_size = rand() % 40;
	//fill the vector with random numbers between 1 and 10
	for(int i = 0; i < random_size;i++){
		random.push_back(rand() % 10);
	}

	//print info of the vector
	cout << "The size of the vector is: " << random.size() << "\n";
	cout << "Vector elements: ";
	for(int i : random)
		cout << i << " ";
	cout << "\n\n";

	//access the first and last element
	int last = random.size()-1;
	cout << "The first element is : " << random[0] << ", the last element is: " << random[last] << "\n\n";
	//swap the first and last element
	int temp = random[0];
	random[0] = random[last];
	random[last] = temp;
	cout << "After swapping, the first element is: " << random[0] << ", the last element is: " << random[last] << "\n";

	return 0;
}
```

Vectors in Functions

- Just like arrays, we can have vectors as parameters.
- Unlike arrays, vectors are passed by value. Meaning an identical copy of that vector is created in the scope of the function.
- This can be very time and resource consuming, so whenever possible, pass your vectors by reference.

Example:

```
void fillVector(vector<int> & v);
```

If you don't want to modify your vector, just add the const keyword

Example:

```
void printVector(const vector<int> & v);
```

Notice that for vectors, we don't have to pass the size, since we always have access to the .size() function.

Let's look some functions with vectors and also user input into a vector. (vectors.cpp)