Standard Template Library

Templates:

In C++ templates are a commonly used tool that provide functionality which is not specific to any particular datatype. Where without the use of templates a class or function may be written multiple times, once for each application involving a different datatype, templates offer a single class or function that can be applied to any datatype. Templates essentially provide code-reuse that in turn greatly reduces the potential programming effort that might otherwise be required. This ability to be independent of any specific datatype is achieved through the use of template parameters which allows a type to be passed as an argument. Templates are usually used to replace functions or classes that contain a structure which can be applied to any datatype.

Standard Template Library:

"The Standard Template Library, or STL, is a C++ library of container classes, algorithms, and iterators; it provides many of the basic algorithms and data structures of computer science" [1] in the form of templates. STL containers are used to store objects just like regular containers but as outlined above they are not confined to any particular data type. There are a number of different container types such as vector, list, deque, set, multiset, map, multimap, hash_set, hash_multiset, hash_map, and hash_multimap all of which offer various types of data storage structures.

Vector Template:

The STL vector container class is one of the most efficient elements in the C++ STL. It offers storage which is variable in size and linear in structure, one of the most powerful attributes of this template is its automated memory allocation. These characteristics allow the STL vector to support "random access to elements, constant time insertion and removal of elements at the end, and linear time insertion and removal of elements at the beginning or in the middle" [2]. This container class can be applied to many of the same applications as an array, but its size is larger than what is actually required to hold its elements in order to allow for possible expansion, this is how it supports dynamic growth.

Application:

The application of the STL vector container class in the Fantasy Game project was to represent the board on which the game is played, so initially when the program starts the board is initialised with the statement "`vector< vector<int> > board::map(10,vector<int>(10));`" this creates a vector of vectors which represents a 2D-array or the board, which is 10 × 10 elements in apparent size. The user is then asked to input a number of rows and columns which will modify the size of the board to suit their requirements, on which the different elements of the game can then be stored allowing each element to be independently manipulated. All perimeter elements of the board are then assigned a value 5 which makes that element a "Wall" or boarder around the board.

References:

[1] (2016) Silicon Graphics International. [Online]. Available: http://www.sgi.com/tech/stl/stl_introduction.html.
[2] (2016) Silicon Graphics International. [Online]. Available: http://www.sgi.com/tech/stl/Vector.html.