Employing vector arrays

A vector is an alternative to a regular array and has the advantage that its size can be changed as the program requires. Like regular arrays, vectors can be created for any data type and their elements are also numbered starting at zero.

In order to use vectors in a program the C++ **vector** library must be added with an **#include <vector>** preprocessor directive at the start of the program. This library contains the predefined functions in the table below, which are used to work with vectors:

Function:	Description:
at(*number* **)**	Gets the value contained in the specified element number
back()	Gets the value in the final element
clear()	Removes all vector elements
empty()	Returns true (1) if the vector is empty, or returns false (0) otherwise
front()	Gets the value in the first element
pop_back()	Removes the final element
push_back(*value* **)**	Adds a final element to the end of the vector, containing the specified value
size()	Gets the number of elements

A declaration to create a vector looks like this:

vector < ***data-type*** > ***vector-name*** **(** ***size*** **) ;**

An **int** vector will, by default have each element automatically initialized with a zero value. Optionally a different initial value can be specified after the size in the declaration with this syntax:

vector < ***data-type*** > ***vector-name*** **(** ***size*** **,** ***initial-value*** **) ;**

The functions to work with vectors are simply appended to the chosen vector name by the dot operator. For example, to get the size of a vector named "vec" you would use **vec.size()** .

Don't forget

Individual vector elements can be referenced using square brackets as with regular arrays, such as **vec[3]**.

...cont'd

vector.cpp

1. Start a new program by specifying the C++ library classes to include and a namespace prefix to use

```
#include <vector>                    // Include vector support.
#include <iostream>
using namespace std ;
```

2. Add a main function containing a final return statement

```
int main()
{
  // Program code goes here.
  return 0 ;
}
```

3. In the main function, insert a statement to declare and initialize a vector array of three elements of the value 100

```
vector <int> vec( 3, 100 ) ;
```

4. Now insert statements to manipulate the vector elements

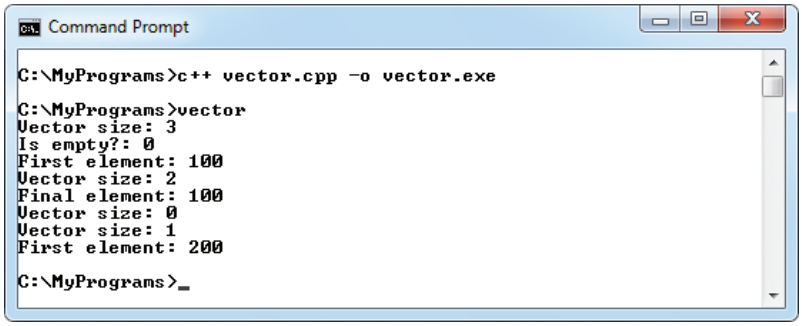
```
cout << "Vector size: " << vec.size() << endl ;
cout << "Is empty?: " << vec.empty() << endl ;
cout << "First element: " << vec.at(0) << endl ;

vec.pop_back() ;            // Remove final element.
cout << "Vector size: " << vec.size() << endl ;
cout << "Final element: " << vec.back() << endl ;

vec.clear() ;               // Remove all elements.
cout << "Vector size: " << vec.size() << endl ;

vec.push_back( 200 ) ;      // Add an element.
cout << "Vector size: " << vec.size() << endl ;
cout << "First element: " << vec.front() << endl ;
```

5. Save, compile, and run the program to see the output



Hot tip

The example on page 50 shows how to use a loop to populate a vector with different initial values in each element.