**Chapter 16**

**Standard Template Library**

**Templates -Generic Functions/Generic Classes**

Templates provide an alternative to inheritance for creating generic functions and generic classes. Users can create their own customized templates - C++ does provide a set of templates called the Standard Template Library (STL).

**Containers - Vectors/Lists**

The STL provides a set of classes used for storing data - 2 most commonly used is the vector and list. Vector is implemented as a dynamically allocated array while a list is implemented as a linked list.

Vector - An expandable array. Values may be added to or removed from the end of the vector efficiently.

list - A doub ly linked list of data elements. Values may be inserted to or removed from

any position efficiently.

The advantage to using these classes is that they can contain any type of data - any class any primitive - without creating a new class for each one. These are *generic* containers - can contain any type.

.

**Vector**

Vector containers are implemented as dynamic arrays; Just as regular arrays, vector containers have their elements stored in contiguous storage locations, which means that their elements can be accessed using offsets on regular pointers to elements.  
  
But unlike regular arrays, storage in vectors is handled automatically, allowing it to be expanded and contracted as needed.  
  
Vectors are good at:

* Accessing individual elements by their position index (constant time).
* Iterating over the elements in any order (linear time).
* Add and remove elements from its end (constant amortized time).

**Declaring a Vector**

Vectors are declared using the special <> notation to indicate the data the vector is to contain:

vector<GenPoint> allPoints;

In this case, the vector is to contain GenPoint. The <> notation is template notation and is required when declaring a variable of a template type.

**Vector methods**

Most relevant vector methods are:

size( ) Returns the number of elements in the vector.

push\_back()Accepts as an argument a value to be inserted into the vector. The argument is insetted after the last element.

pop\_back ( ) Removes the last element from the vector.

operator [ ] Allows array-like access of existing vector elements, (The vector must already

contain elements for this operator to work. It cannot be used to insert values into the vector.)

**see example vector1.cpp.**