**Container classes**

**What is a container class? What are the types of container classes?**

Container class is a class that hold group of same or mixed objects in memory. It can be heterogeneous and homogeneous. Heterogeneous container class can hold mixed objects in memory whereas when it is holding same objects, it is called as homogeneous container class.

* container class which is utilized for the purpose of holding objects in memory
* The purpose of container class is to hide the topology for the purpose of objects list maintenance in memory.

**What are the C++ standardized container classes?**

The following are the standardized container classes :  
**1. std::map :**   
Used for handle sparse array or a sparse matrix.  
**2. std::vector :**  
Like an array, this standard container class offers additional features such as bunds checking through the at () member function, inserting or removing elements, automatic memory management and throwing exceptions.   
**std::string :**   
A better supplement for arrays of chars.

**Container class functions?**

* Create an empty container (via a constructor)
* Insert a new object into the container
* Remove an object from the container
* Report the number of objects currently in the container
* Empty the container of all objects
* Provide access to the stored objects
* Sort the elements (optional)

**Difference between Inheritance and Containership**

* Inheritance is the ability for a class to inherit properties and behaviour from a parent class by extending it, while Containership is the ability of a class to contain objects of different classes as member data.
* If a class is extended, it inherits all the public and protected properties/behaviour and those behaviours may be overridden by the subclass. But if a class is contained in another, the container does not get the ability to change or add behaviour to the contained.
* Inheritance represents an “is-a” relationship in OOP, while Containership represents a “has-a” relationship.

**Direct and indirect base classes in C++**

|  |  |
| --- | --- |
| **Direct** | **Indirect** |
| A base class can be defined classified into two types, direct bas and indirect base.  A base class is called a direct base if it is mentioned in the base list.  1. *Class base A*  {  —————–  ——————  };  Class derived B: public base A  {  ———————–  ———————–  };  2.*Class base B*  {  —————  —————–  };  Class base B  {  —————-  —————  };  Class derived C: public base A, public base B  {  —————-  —————–  };    Where both classes base A and B are the direct base. | A derived class can itself serve as a base of another class. When a derived class is declared as a base of another class, the newly derived class inherits the properties of its base classes as well as its data and function member also.  A class is called indirect base class if it is not a direct base but is a base class of one of the classes mentioned in the above list.  Class base A  {  —————-  —————-  };  Class derived B: public base A  {  —————  —————  };  Class derived: public derived B  {  ——————-  ——————–  };  Where, base A is an indirect base for derived C. |