CHAPTER 9: TEMPLATES

[5HOURS].

DEFINITION OF TEMPLATE:

- **Templates** are powerful features of **C++** which allows you to write generic programs.
- In general, you can create a single function or a class to work with different data types using **templates**.
- **Templates** are often used in larger codebase for the purpose of code reusability and flexibility of the programs.
- Templates are used to prevent us from writing the same function or class separately for different data types.
- Using templates, we achieve freedom from data types.

- There are two types of templates in C++.

❖ **Function Templates(also known as generic function):**

- Function Templates prevent us from defining separate functions performing the same task for different data types.

Syntax: template <class type> type func_name (type arg1,……){}

❖ **Class Templates(also known as generic class):**

- Similar to function template, we can declare class template that operate on any types of data.
- A class that operates on any type of data is called class template.

Syntax: template <class type> class class_name { //…body of class};

- Once a class template is declared , we create a specific instance of the class using following **syntax: class_name <data_type1, data_type2,…….> object;**

OVERLOADING FUNCTION TEMPLATE:

- Overloading with function.
- Overloading with other template.

[**We will program for each on IDE.]

CLASS TEMPLATE :

- **Function Definition of class Template:**

The member function of class template defined in outside class can be done in following way:

```
template<class type>
  class class_name{
    return_type  function_name (type arg);
    //body……..} ;
template < class  type>
 return_type  class_name <type>::function_name (type arg){//body}
```

[We have to cover following topics too:]

- Non-Template type arguments.
- Default Argument with class Template.

❖ **Derived Class Template:**

- Similar to the inheritance in normal class, the class template can also be inherited . Inheritance with class template provides a mechanism for building new data types from existing ones which has some form of common behaviors and properties.
- When creating derived class with template mechanism, we can create derived class with following ways:

1. We can create a derive class(which is a non template)from base class (which is a template).
2. We can create a derived class which is template from a base class which is also a template with the same template parameters as in the base class.
3. We can create a derive class which is a template from a base class which is also a template with additional template parameters in the derived class than that of the base class.
4. We can create a derived class (which is a template) from a base class (which is not a template).

[**We will program for all these on IDE.]

INTRODUCTION TO STANDARD TEMPLATE LIBRARY:

- A collection of generic class and functions is called the Standard Template Library(STL).
- It is a powerful set of C++ templates classes.
- The STL contains several components. But its core are following three key components:

1. Containers.
2. Algorithms.
3. Iterators.

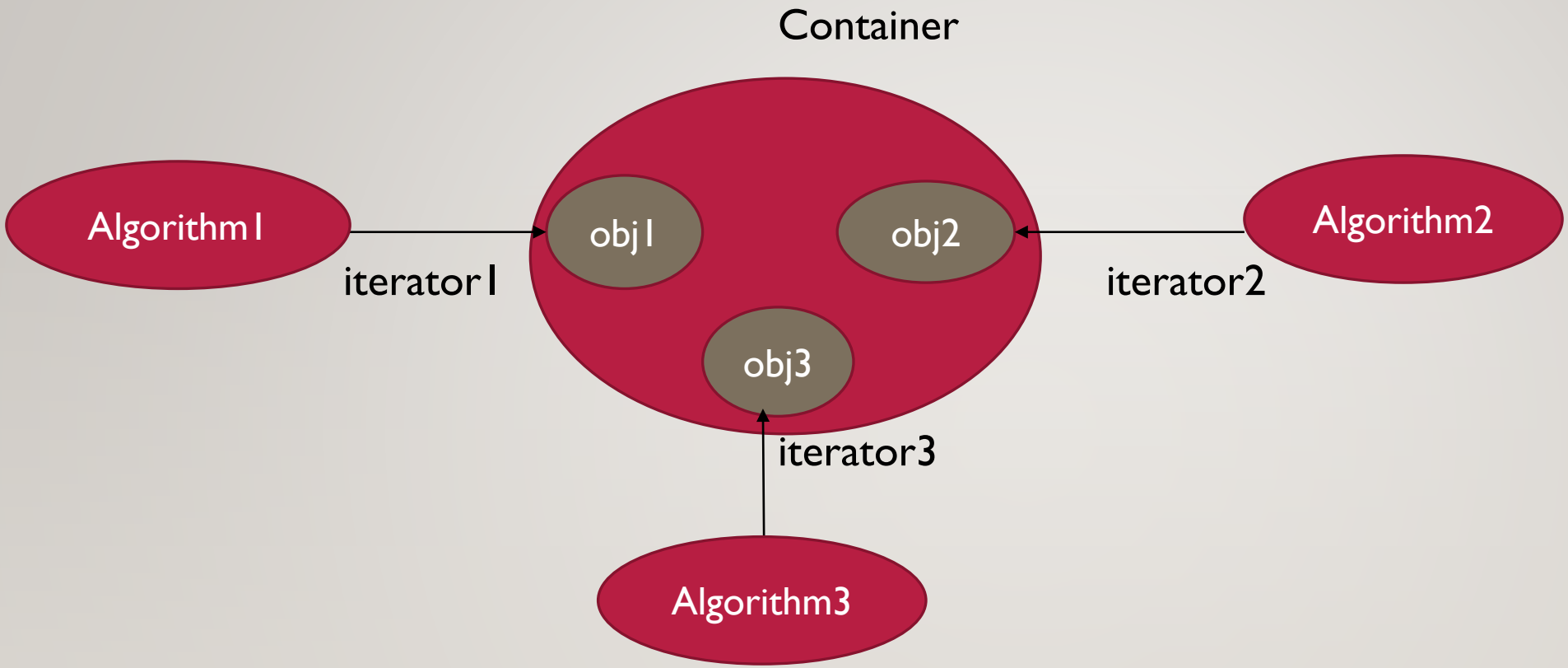

fig: Relationship between three component.

1.CONTAINER:

- The STL Containers are data structure (class) capable of storing objects of any data type in an organized way in memory.
- It is the way data is organized in a memory .
- STL containers are implemented as template classes and therefore can be easily customized to hold different types of data.
- STL defines ten containers that are grouped into three categories:

1. Sequence Container.
2. Associative Container.
3. Derived Container.

1. **Sequential Container**:

- Sequence Container is a variable sized container whose elements are arranged in a strict linear order.
- It stores elements in a linear list or manner.
- Each elements is related to one other elements by its position along the line.
- They are categorize into following types:

1. Vector(dynamic array).
2. List
3. Deque

2. Associative Container:

- Associative Containers are variable-sized containers that supports efficient retrieval of elements based on keys.
- A containers which allows efficient retrieval of value based on keys.
- They are not sequential.
- There are four types of associative containers :

1. Set
2. Multiset
3. Map
4. Multimap

- All these containers store data in a structure called that facilitates fast searching , deletion and insertion.
- Each element in this container has key value.

Note:

- The **map** and the **multimap** are both containers that manage key/value pairs as single components. The essential difference between the two is that in a **map** the keys must be unique, while a **multimap** permits duplicate keys.
- Containers set and multiset can store number of items containing keys . The main difference in a set and multiset is that a multiset allows duplicate items while a set does not.

3. Derived Containers (Container Adaptors):

- These containers are derived from sequential container.
- STL provides three derived containers namely:
- **Stack(Where elements are inserted ,removed in LIFO (last in first out)).**
- **Queue(follows FIFO(first in first out)).**
- **Priority_queue(Where insertion ,removal and traversal is performed on the top elements where it guarantee that the top element is the element with highest priority queue) .**

ALGORITHM:

- An algorithm is a procedure that is used to process the data contained in the containers.
- STL provides more than 60 standard algorithms to support more extended or complex operations.
- Standard algorithm also permit us to work with two different type of containers

  at the same time.
- STL algorithm are not member function or friend of containers .They are standalone template function.

STL algorithm based on the nature of operation they perform may be categorize as:

1. Mutating Algorithms.
2. Sorting Algorithms.
3. Set Algorithms.
4. Relational Algorithms.
5. Retrieve or Non-Mutating Algorithms.

ITERATORS:

- Iterators behaves like a pointer and are used to access individual elements in container.
- They are often used to traverse from one elements to another , a process known as iterating through the container.
- That means if the iterator point to the one elments in range then it is possible to increase or decrease iterator so that we can access next elements in the range.
- Iterator connects algorithm with container and play a key role in the manipulation of data stored in the containers.

There are five type of iterators. They are:

Iterators	Access Method	Direction of movement	I/O capability	Remarks
Input	Linear	Forward Only	Read only	Cannot be saved
Output	Linear	Forward Only	Write only	Cannot be saved
Forward	Linear	Forward Only	Read/Write	Can be saved
Bidirectional	Linear	Forward and Backward	Read/Write	Can be saved
Random	Random	Forward and Backward	Read/Write	Can be saved

Note:

- Only sequential and associative containers are transferrable with iterator.
- The input and output support the least function .They can be used only to traversal in a container.
- The forward iteration supports all operations of input and output and also retains its position in the containers.
- Bidirectional supports two way traversal i.e. forward and backward.
- The random access iterator combines the functionality of bidirectional and ability to jump to an arbitrary location.

1. Vector:

- Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the containers.
- Vector supports random access and a wide range of iteration operations.
- The indexing and iteration are lighting –fast , being basically the same as indexing and iterating over an array of objects.
- It contains number of member function like insertion, removal of an element at last ,begining or middle.

- Some Function of vector are listed below:

1. size(): Returns the number of elements in the vector.
2. push_back():It push the elements into a vector from the back.
3. pop_back(): It is used to pop or remove elements from a vector from the back.
4. insert() – It inserts new elements before the element at the specified position
5. erase():It is used to remove elements from a container from the specified position or range
6. empty(): Returns whether the container is empty.
7. back(): Returns a reference to the last element in the vector

LIST:

- It is doubly linked list.
- It supports a bidirectional , linear list and provides an efficient implementation for deletion and insertion operation.
- It does not supports random access so it can be accessed sequentially only.
- Since list stores address of front and back elements , it can be accessed from both ends.
- Some of its member function are :

- push_front(): Add an elements to end.
- push_back() : Add an elements to front.
- begin(): Give reference to first elements.
- clear() : Deletes all the elements.
- pop_front(): Deletes first elements.
- pop_back(): Deletes last elements.
- reverse() : reverse the list.
- size(): give the size of list.
- merge() :merge two ordered list.

MAPS:

- A map is sequence of (key,value) pairs where a single value is associated with each unique key.
- It is used as container that resembles an array. But instead of accessing its elements based with index number , we can access based on the key and is very fast.
- We should specify a key to obtain the associated value.
- Some function are:

- begin() : Give reference to first elements.
- empty(): Determine map is empty or not.
- clear() : Deletes all the elements.
- size(): Give the size of the list.
- end(): Give reference to the end of the map.
- find(): Give the location of the specified elements.