IDHAYA COLLEGE FOR WOMEN

KUMBAKONAM – 612 001

PG & RESEARCH DEPARTMENT

OF

COMPUTER SCIENCE

ACADEMIC YEAR	:	2019-2020
SEMESTER	:	II
CLASS	:	I B. SC (CS)
SUBJECT INCHARGE	:	R. KANNIKA DEVI
SUBJECT	:	PROGRAMMING IN C++
SUBJECT CODE	:	16SCCCS2

UNIT – V

STANDARD TEMPLATE LIBRARY

Standard Template Library, Manipulating Strings, Object Oriented Systems Development.

Standard Template Library (STL):

It is a set of C++ template classes that provide generic classes and function that can be used to implement data structures and algorithms.

Generic Functions or Classes?

In many times while programming, there is a need for creating Functions which perform the same operations but work with different data types. So C++ provides a feature to create a single generic function instead of many functions which can work with different data type by using the template parameter.

Syntax:

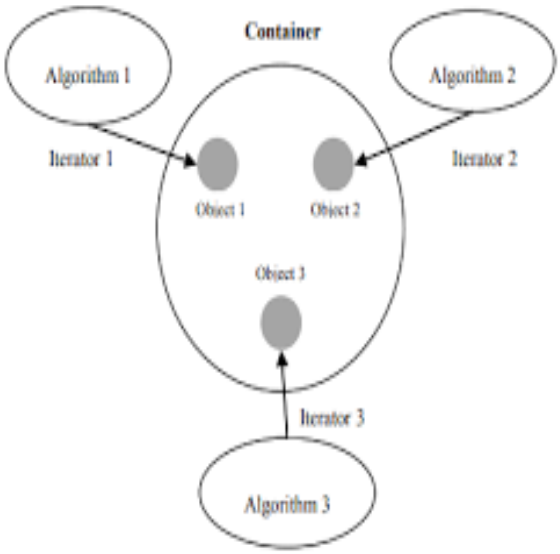
Template < class type> return_type fun_name(parameter-list)

'Type' is just a placeholder used to store the data type when this function is used you can use any other name instead class is used to specify the generic type of template, alternatively typename can be used instead of it.

Components of STL:

The STL contains several components. They are

- Containers
- Algorithms
- Iterators



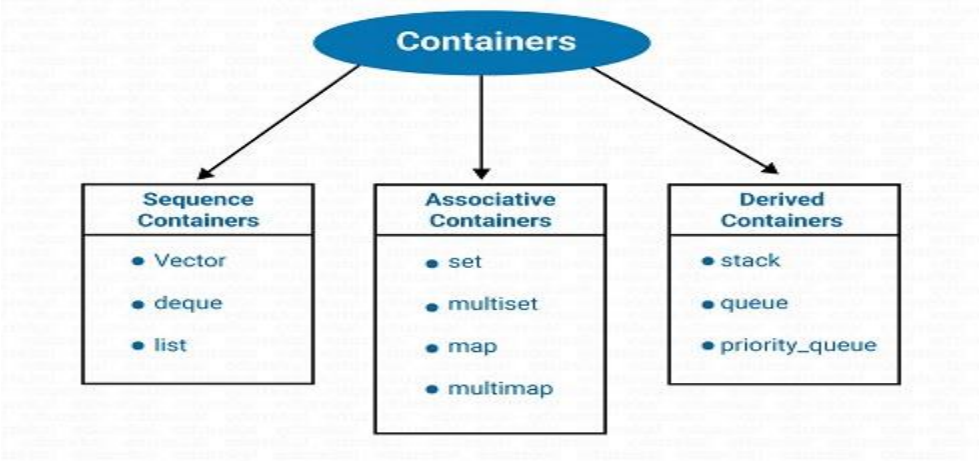
Containers:

Container is an object that actually stores data. It is a way data is organized in memory. STL containers are implemented by template classes.

It helps us to implement and replicate simple and complex data structures very easily like arrays, list, trees, associative arrays and many more.

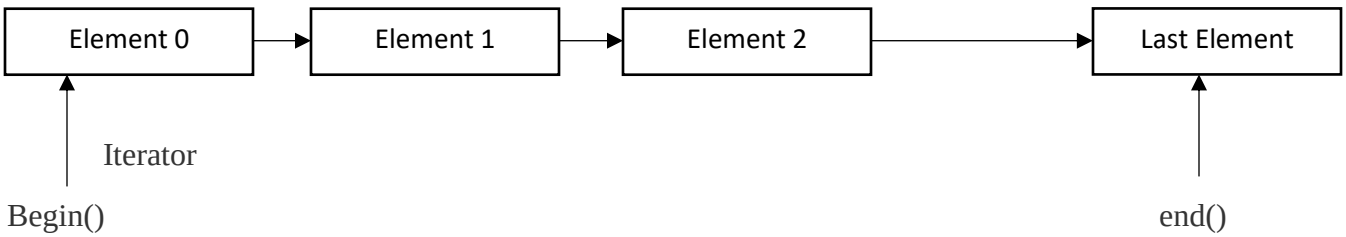
Following are some common containers:

- **vector**: replicates arrays
- **queue**: replicates queues
- **stack**: replicates stack
- **priority queue**: replicates heaps
- **list**: replicates linked list
- **set**: replicates trees
- **map**: associative arrays



Sequence containers:

Sequence containers store elements in a linear sequence. Each element is related to other elements by its position along the line.



Associative containers:

- **Associative containers** are designed to support direct access to elements using keys.
- It used to implement sorted data structures such as map, set etc.

Set and Multiset:

- It can store a number of items and provide operations for manipulating them using the values as the key.
- The main difference between set and multiset is that a multiset allows duplicate items while a set does not.

Map and Multimap:

- It used to store pairs of items, one called the Key and the other called the Value.
- The main difference between a map and multimap is that a map allows only one key for a given value to be stored while multimap permits multiple keys.

Derived Containers:

- It provides three containers namely Stack, Queue and Priority queue. These are known as Container Adaptors.
- It support two member functions pop() and push() for implementing deleting and inserting operations.

Algorithms:

- In STL are procedures that are applied to containers to process their data. It provides more than sixty standard algorithms to support more extended or complex operations.
- STL algorithms are not member functions or friends of containers.

They can be categorized as follows:

- Retrieve algorithms
- Mutating algorithms
- Sorting Algorithms
- Set and Relational algorithm, Mutating algorithms

Operations	Description
Copy()	Copies a sequence
Copy_backward()	Copies a sequence from the end
Fill()	Fill a sequence with a specified value
Fill_n ()	Fill first n elements with a specified value
Generate()	Replaces all elements with the result of an operation

Iterators:

Iterators are used to point at the memory addresses of STL containers. They are primarily used in sequence of numbers, characters etc. They reduce the complexity and execution time of program.

Iterator Types

	Output	Input	Forward	Bi-directional	Random
Read		`x = *i`	`x = *i`	`x = *i`	`x = *i`
Write	`*i = x`		`*i = x`	`*i = x`	`*i = x`
Iteration	`++`	`++`	`++`	`++, --`	`++, --, +, -, +=, -=`
Comparison		`==, !=`	`==, !=`	`==, !=`	`==, !=, <, >, <=, >=`

- **Output**: write only and can write only once
- **Input**: read many times each item
- **Forward**: supports both read and write
- **Bi-directional**: supports also decrement
- **Random**: supports random access (just like C pointer)

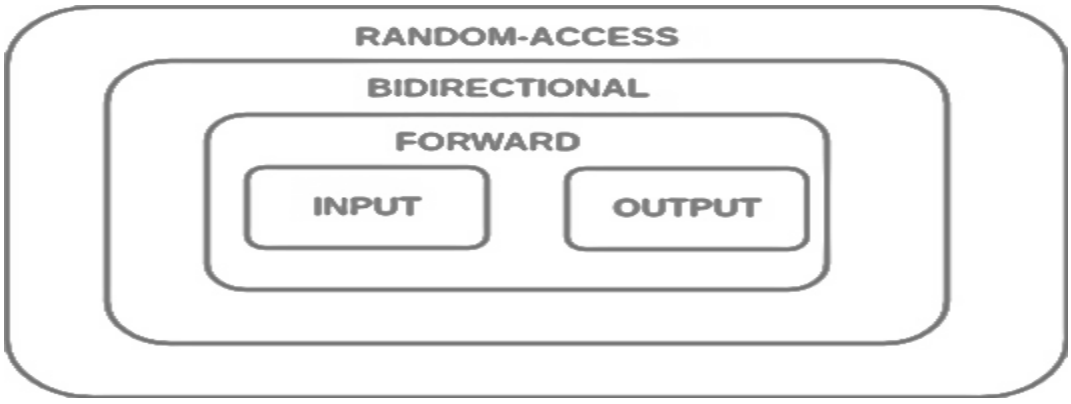


Fig. Functionality Venn diagram of iterators

(i) The input and output iterators support the least functions. They can be used to traverse in a container.

(ii) The forward operator supports all operations of input and output iterators and also retains its position in the container.

(iii) A bidirectional iterator supporting all forward iterator operations provides the ability to move in the backward direction in the container.

(iv)A random-access iterator combines the functionality of a bidirectional iterator with an ability to jump to an arbitrary location.

Application of Container Classes:

The three most popular containers namely **Vector, List and Map.**

Vectors:

The vector is the most widely used container.It stores elements in contiguous memory locations and enables direct access to any element using the subscript operator[].

Ex:

```
Vector<int> v1;              //zero-lemgth int vector
Vector<double> v2(10);       //10-element double vector
```

Function	Task
At ()	Gives a reference to an element
Back ()	Gives a reference to the last element
Begin ()	Gives a reference to the first element
Capacity ()	Gives the current capacity of the vector
Clear ()	Deletes all the elements from the vector

Lists:

- The list supports a bidirectional, linear list and provides efficient implementation for deletion and insertion operations.
- The objects of list can be initialized with other list objects like,

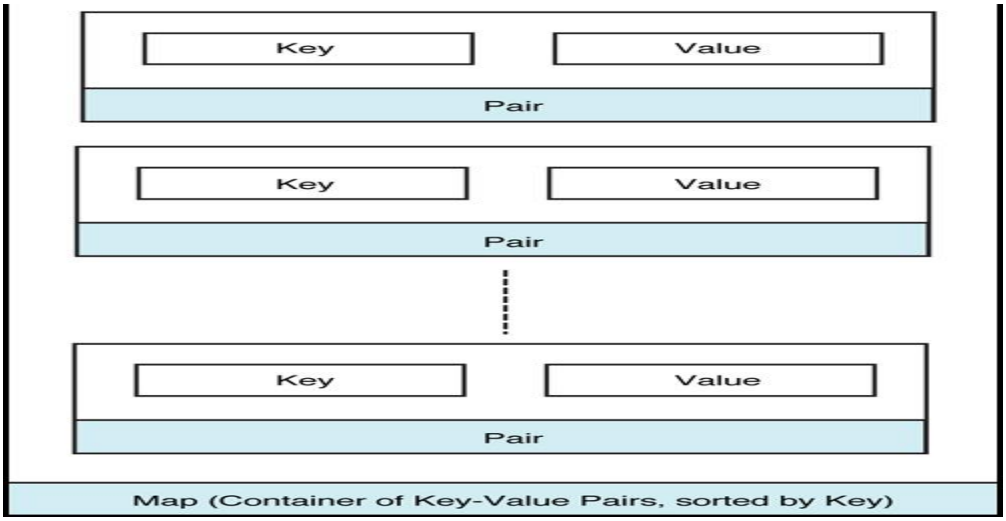
List A = List1;

List B = List2;

The statement, List1.merge(list2);

Maps:

A Map is a sequence of (key, value) pairs where a single value is associated with each unique key. Retreival of values is based on the key and is very fast.



Some basic functions associated with Map:

begin() – Returns an iterator to the first element in the map
end() – Returns an iterator to the theoretical element that follows last element in the map
size() – Returns the number of elements in the map

max_size() – Returns the maximum number of elements that the map can hold
empty() – Returns whether the map is empty
pair insert (keyvalue, mapvalue) – Adds a new element to the map
erase(iterator position) – Removes the element at the position pointed by the iterator

Function objects:

A function object is a function that has been wrapped in a class. The class has only one member function the overloaded () operator and no data.

Ex:

Sort (array, array+5, greater<int>());

Greater<int> () to sort the elements contained in array in descending order.

Manipulating String:

- A string is a sequence of characters. C++ now provides a new class called string. For using the string class, we must include <string> in program.
- The string class is very large and includes many constructors, member functions and operators.
- Creating string objects
- Reading string objects
- Displaying string objects to the screen
- Modifying string objects

- Comparing string objects
- Accessing characters in a string

Constructor	Usage
String ()	For creating an empty string
String (const char *str);	For creating s string object from a null-terminated string
String (const char & str);	For creating s string object from other string object

Table. Commonly used string constructors

The important functions supported by the string class are Append (), Assign (), At (), Begin (), Capacity (), Compare (), length (), replace (), size (), swap ().

Operator	Meaning
=	Assignment
+	Concatenation
+=	Concatenation assignment
!=	Inequality
<	Less than
<=	Less than or equal

Creating String Objects:

We can create string objects in a number of ways. they are

```
String s1;          //Using constructor with no argument

String s2("xyz");   //Using one-argument constructor

s1=s2;              //Assigning string objects

getline(cin, s1);   //Reading through keyboard a line of text
```

The overloaded operator concatenates two string objects. We can also use the operator += to append a string to the end of the string.

Ex:

```
s3 += s1;

s3 += "abc";
```

Manipulating string objects:

We can modify contents of string objects in several ways, using member functions such as insert (), replace (), erase () and append ()

```
#include<iostream>

#include<string>

int main ()

{

String s1(“12345”);

String s2(“abcde”);

Cout<<” Original strings are:\n”;

Cout<<” s1: “<<s1<<” \n”;

Cout<<” s2:” <<s2<<” \n\n”;

Cout<<” place s2 inside s1”;

s1. insert (4, s2);

Cout<<” modified s1: “<<s1<<” \n\n”;

Cout<<” remove 5 characters from s1\n”;

s1. erase (4,5);

Cout<<” now s1: “<<s1<<” \n\n”;

Cout<<replace middle 3 characters in s2 with s1\n”;

s2. replace (1,3, s1);

Cout<<” now s2: “<<s2<<” \n”;

Return 0;

}
```

Accessing characters in strings:

The string class supports the following functions are as follows:

At() - For accessing individual characters

Substr() - For retrieving a substring

Find () - For finding a specified substring

Find_first_of() - For finding the location of first occurrence of the specified character(s)

Find_last_of () - For finding the location of last occurrence of the specified character(s)

Object-Oriented Systems Development:

Software Development:

The software development process consists of Planning, Analysis, Design, Development and maintenance. The software development procedures integrate the methods and tools.

A successful system must

- Satisfy the user requirements
- Be easy to understand by the users and operators
- Be easy to operate
- Be easy to modify
- Be expandable
- Handle the errors and exceptions satisfactorily
- Be delievered on schedule within the budget

Procedure-oriented Paradigms:

A program in a procedural language is a list of instruction where each statement tells the computer to do something. It focuses on procedure (function) & algorithm is needed to perform the derived computation.

When program become larger, it is divided into function & each function has clearly defined purpose. Dividing the program into functions & module is one of the cornerstones of structured programming.

E.g.: - c, basic, FORTRAN.

Characteristics of Procedural oriented programming: -

1. It focuses on process rather than data.
2. It takes a problem as a sequence of things to be done such as reading, calculating and printing. Hence, a number of functions are written to solve a problem.
3. A program is divided into a number of functions and each function has clearly defined purpose.
4. Most of the functions share global data.
5. Data moves openly around the system from function to function.

The activities are as follows:

Problem Definition:

This activity requires a precise definition of the problem in user terms.

Analysis:

This covers a detailed study of the requirements of both the user and the software. This activity is basically concerned

What are the inputs to the system?

What are the process required?

What are the outputs expected?

Design:

The design phase deals with various concepts of system design such as data structure, software architecture and algorithms.

Coding:

Coding refers to the translation of the design into machine-readable form.

Testing:

Once the code is written, it should be tested for correctness of the code and results.

Maintenance:

It occurs due to a change in the user's requirement, a change in the operating environment, or an error in the software that has not been fixed during the testing. It ensures these changes are incorporated wherever necessary.

Procedure-oriented development tools:

- The development tools available today may be classified as the first generation, second generation and third generation.
- The first-generation tools developed in 1960s and 1970s are called Traditional tools.
- The second-generation tools introduced in the late 1970s and early 19080s meant for structured systems analysis and design. They are known as Structured tools.
- The recent tools are third generation evolved since late 1980s to object-oriented analysis and design.

System Flowcharts:

A graphical representation of the important inputs, outputs, and data flow among the key points in the system.

Program flowcharts:

A graphical representation of the program logic.

Playscripts:

A narrative description of executing a procedure.

Layout forms:

A format designed for putting the input data or displaying results.

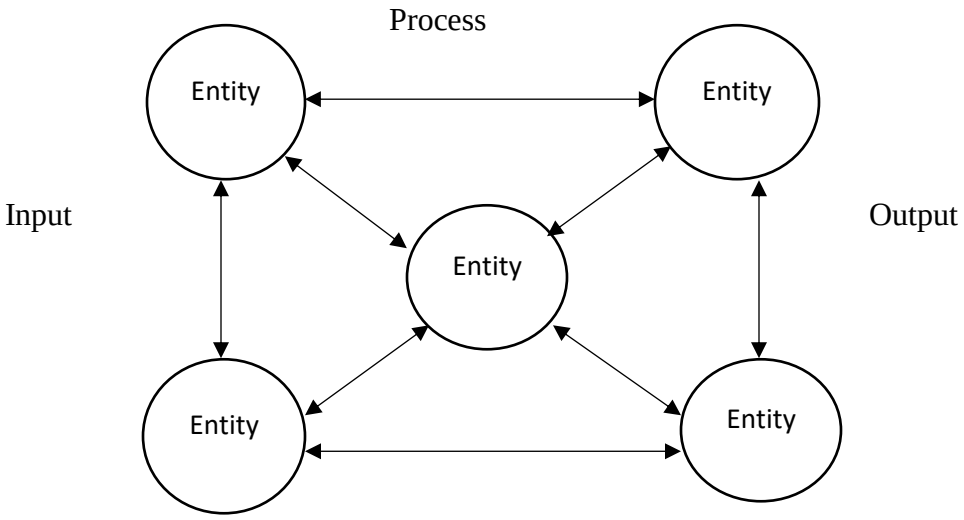
Grid charts:

A chart showing the relationship between different modules of a system

Object Oriented Paradigm:

- The object-oriented paradigm draws on the general systems theory as a conceptual background. A system can be viewed as a collection of Entities that interact together to accomplish certain objectives.

- Entities may represent physical objects such as equipment and people and abstract objects such as data files and functions. Entities are called objects.



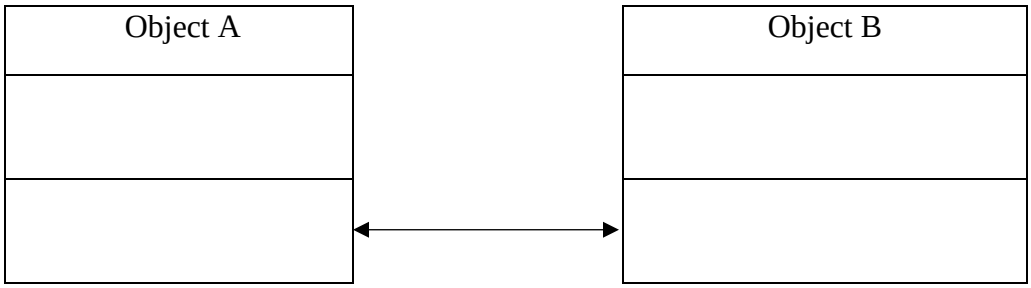
- Object oriented analysis (OOA) refers to methods of specifying requirements of the software in terms of real-world objects, their behavior, and their interactions.
- Object-Oriented Programming (OOP) refers to the implementation of the program using objects in an object-oriented programming language.

Object-Oriented Notations and Graphs:

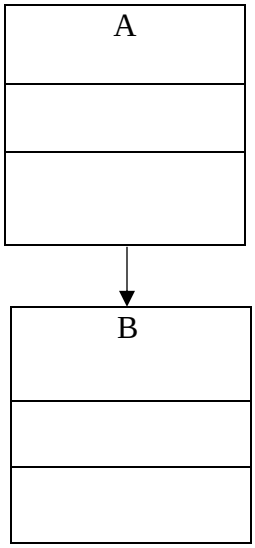
Graphical Notations are an essential part of design and development process and object-oriented design.

It represents classes, objects, subclasses, and their inter-relationships. Authors and notations have used their own notations.

- Classes and objects
- Instances of objects
- Message communication between objects
- Inheritance relationship
- Classification relationship
- Composition relationship
- Hierarchical chart
- Client-server relationship
- Process layering



(Message Communication between objects)



Base class

Derived class

Inheritance Relationship

Steps in Object-Oriented Analysis:

The Object-oriented Analysis (OOA) approach consists of the following steps:

- Understanding the problem
- Requirements Specification
- Identification of Objects
- Data Flow Diagram
- Textual Analysis

Problem Understanding:

The first step in the analysis process is to understand the problem of the user. The problem statement should be stated in a single grammatical correct sentence.

Requirements Specification:

This stage to generate a list of user requirements. A clear understanding should between the user and the developer.

Identification of objects:

The application may be analyzed by using the following approaches:

- Data Flow Diagrams (DFD)
- Textual Analysis (TA)

Data Flow Diagram:

It indicates how the data moves between from one point to another in the system.

Textual Analysis:

The description may be of one or two sentences or one or two paragraphs depending on the type and complexity of the problem.

Steps in object-oriented design:

The object-oriented design (OOD) approach may involve the following steps:

- Review of objects created in the analysis phase
- Specification of class dependencies
- Organization of class hierarchies
- Design of classes
- Design of member functions
- Design of driver program

Prototyping Paradigm:

- A prototype is a scaled down version of the system and may not have performance criteria and resource requirements.
- Protype provide an opportunity to experiment and analyze various aspects of the system such as system structure, Internal design, Hardware requirements.

The prototype paradigm are as follows:

- System specifications
- Outline requirements
- Design prototype model
- Build prototype
- Evaluate prototype
- Male detailed
- Full system