CORE COURSE II

PROGRAMMING IN C++

Unit I

Basic Concepts of Object- Oriented Programming - Benefits of OOP - Object Oriented Languages - Applications of OOP – Structure of C++ Program - Tokens, Expressions and Control Structures – Functions in C++

Unit II

Classes and Objects – Constructors and Destructors –Operator Overloading and Type Conversions

Unit III

Inheritance : Extending Classes – Pointers - Virtual Functions and Polymorphism

Unit IV

Managing Console I/O Operations – Working with Files – Templates – Exception Handling

Unit V

Standard Template Library – Manipulating Strings – Object Oriented Systems Development

PART-A

1. What is STL?

The Standard Template Library (**STL**) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators. It is a generalized library and so, its components **are** parameterized.

2. What are the structured components of STL?

The **Standard Template Library** (**STL**) is a software library for the C++ programming language that influenced many parts of the C++ Standard Library. It provides four components called

- ***algorithms***
- ***containers***
- ***functions***
- ***iterators***

3. Define containers and its types.

A container is an object that actually stores data. It is a way data is organized in memory.The STL containers are implemented by template classes and therefore can be easily customized to hold different types of data.

The three **types** of **containers** found in the **STL** are

- **Sequential**
- **Associative**
- **Unordered**

4. What are the types of derived containers?

The STL provides three types of derived containers

1) Stack
2) Queue
3) Priority queue

5. Define algorithm and its types.

An algorithm is a set of self contained sequence of instructions or actions that contains finite space or sequence and that will give us a result to a specific problem in a finite amount of time.

It is a logical and mathematical approach to solve or crack a problem using any possible method.

Types of algorithm

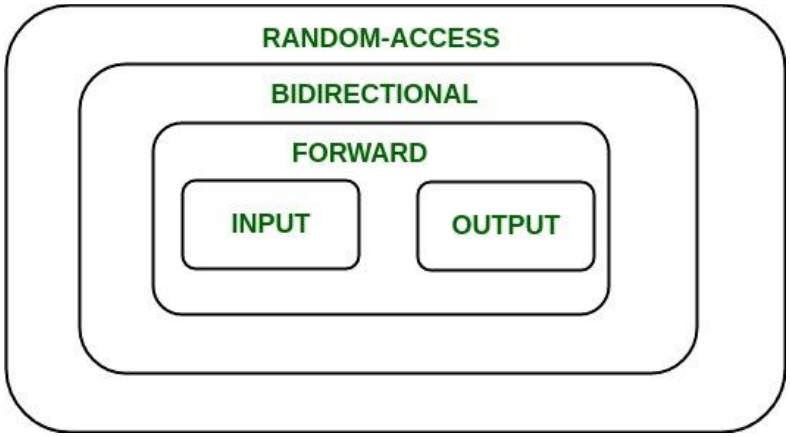
Well there are many types of algorithm but the most fundamental types of algorithm are:

1. Recursive algorithms
2. Dynamic programming algorithm
3. Backtracking algorithm
4. Divide and conquer algorithm
5. Greedy algorithm
6. Brute Force algorithm
7. Randomized algorithm

6. Define iterators and its characters.

An **iterator** is an object (like a pointer) that points to an element inside the container. In use iterators to move through the contents of the container.

The most obvious form of iterator is a pointer. A pointer can point to elements in an array, and can iterate through them using the increment operator (++).



7.What is container adopters?

- Container adapters adapt existing containers to provide new containers.
- In simple terms, STL extension is done with composition instead of inheritance.
- STL containers can't be extended by inheritance, as their constructors aren't virtual.

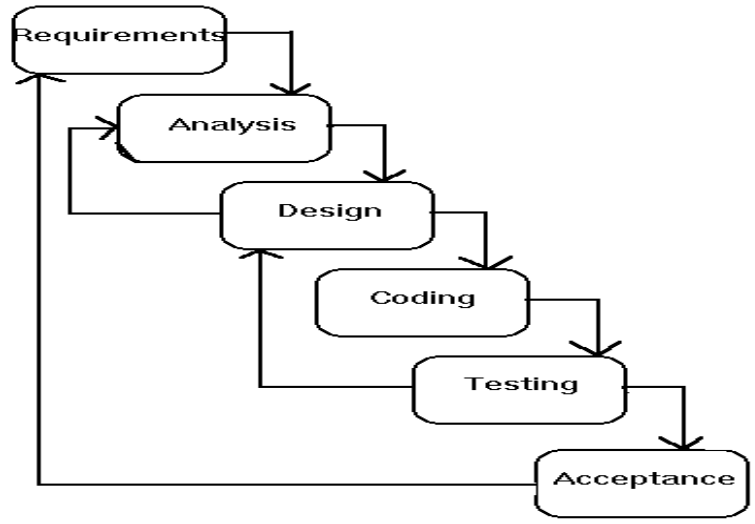
8.Write commonly used string constructors.

Constructor	Description
string(const char *cs)	creates a **string** and initializes it to cs
string(int n, char c)	creates a **string** and initializes it to ns c
string(const char *cs, int n)	creates a **string** and initializes it to the first n elements in cs; if n **is** too big the **string** will contain garbage data

9. What is prototyping?

A **prototype** is an early sample, model, or release of a product built to test a concept or process. It is a term used in a variety of contexts, including semantics, design, electronics, and software programming. **Prototyping** serves to provide specifications for a real, working system rather than a theoretical one.

10.Draw the Classic software development life cycle?



The waterfall model (Systems Development Life Cycle)

11. What are the steps involved in object- oriented design?

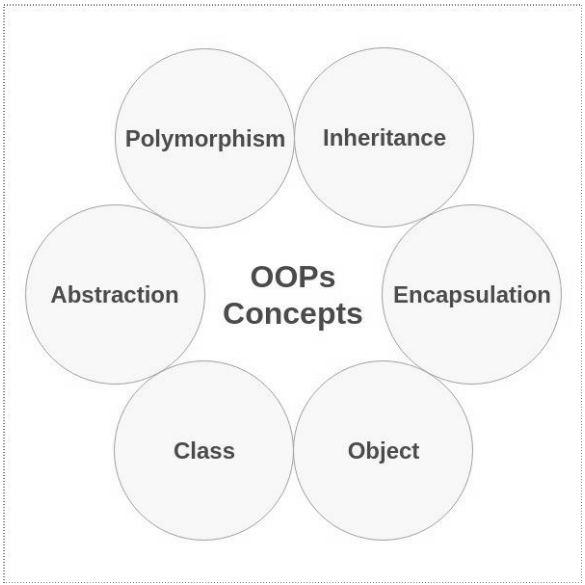
To **design** classes and their attributes, methods, associations, structure, and even protocol, **design** axiom is applied.

- **Design** the access layer.
- Identify access layer class relationship.
- Simplify classes and their relationships.
- Iterate and refine again.
- **Design** the view layer classes.

PART-B

1. Write short notes on oops notations.

Object-oriented programming – As the name suggests uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

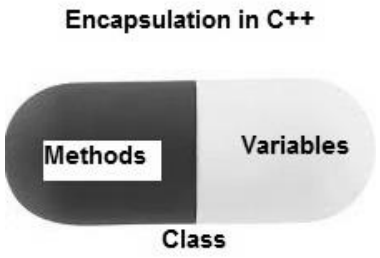


Class: The building block of C++ that leads to Object-Oriented programming is a Class. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

- A Class is a user-defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions define the properties and behaviour of the objects in a Class.
- In the above example of class Car, the data member will be speed limit, mileage etc and member functions can apply brakes, increase speed etc.

Object: An Object is an identifiable entity with some characteristics and behaviour. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

Encapsulation: In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.



Encapsulation also leads to *data abstraction or hiding*. As using encapsulation also hides the data. In the above example, the data of any of the section like sales, finance or accounts are hidden from any other section.

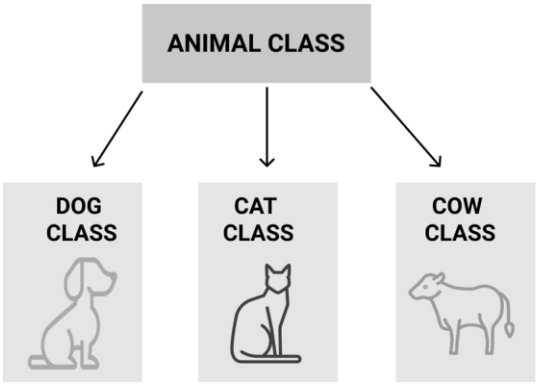
Abstraction: Data abstraction is one of the most essential and important features of object-oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

Polymorphism: The word polymorphism means having many forms. In simple words, In define polymorphism as the ability of a message to be displayed in more than one form.
A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism.

Inheritance: The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.

Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.

Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.



2.Write short notes on STL.

The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators. It is a generalized library and so, its components are parameterized. A working knowledge of template classes is a prerequisite for working with STL.

STL has four components

- Algorithms
- Containers
- Iterators

Algorithms:

The header algorithm defines a collection of functions especially designed to be used on ranges of elements.They act on containers and provide means for various operations for the contents of the containers.

- Sorting
- Searching
- Important STL Algorithms
- Useful Array algorithms
- Partition Operations

Containers

Containers store objects and data. They are basically template-based generic classes.

Containers types :

1) Sequential Containers

Containers the can be accessed in a sequential or linear manner are said to be sequential containers.

Arrays, Vectors, Lists, Deques are the STL containers that store data linearly and can be accessed in a sequential manner.

2) Associative Containers

Associative containers are containers that implement sorted data structures. These containers are fast to search.

3) Container Adopters

Container adopters are sequential containers, however, they are implemented by providing a different interface. Thus containers like a queue, deque, stack, and priority-queue are all classified as container adopters.

Iterators
Iterators are constructs that use to traverse or step through containers in STL. Iterators are very important in STL as they act as a bridge between algorithms and containers. Iterators always point to containers and in fact algorithms actually, operate on iterators and never directly on containers.

Iterators are of following types:

- **Input Iterators:** Simplest and is used mostly in single-pass algorithms.
- **Output Iterators:** Same as input iterators but not used for traversing.
- **Bidirectional Iterators:** These iterators can move in both directions.
- **Forward Iterators:** Can be used only in the forward direction, one step at a time.
- **Random Access Iterators:** Same as pointers. Can be used to access any element randomly.

3. Explain Container supported by the STL?

Containers Library in STL gives us the Containers, which in simplest words, can be described as the objects used to contain data or rather collection of object. Containers help us to implement and replicate simple and complex data structures very easily like arrays, list, trees, associative arrays and many more.

The containers are implemented as generic class templates, means that a container can be used to hold different kind of objects and they are dynamic in nature!

Following are some common containers :

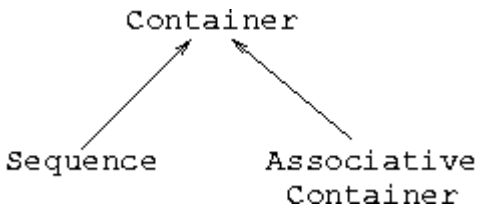
- **vector** : replicates arrays
- **queue** : replicates queues
- **stack** : replicates stack
- **priority_queue** : replicates heaps
- **list** : replicates linked list
- **set** : replicates trees
- **map** : associative arrays
- Classification of Containers in STL

Containers are classified into four categories :

- **Sequence containers** : Used to implement data structures that are sequential in nature like arrays(array) and linked list(list).
- **Associative containers** : Used to implement sorted data structures such as map, set etc.
- **Unordered associative containers** : Used to implement unsorted data structures.
- **Containers adaptors** : Used to provide different interface to the sequence containers.

Container concepts

There are three main container concepts:



Container concepts are not as important for generic programming as iterator concepts.

- There are fewer models.
- Containers have important properties that are not described by the basic container concepts.
  - Properties that differentiate one container from another.
  - In contrast, an iterator is almost fully described by the most refined concept it models.
  - More refined container concepts would have just one model.
- Even the basic concepts can be too refined:
  - boost::array
- There are almost no generic algorithms taking a container as an argument.
  - insert iterators

4.Explain how to manipulate string objects?

A string is a sequence of character. As you know that C++ does not support built-in string type, you have used earlier those null character based terminated array of characters to store and manipulate strings. These strings are termed as *C Strings*. It often becomes inefficient performing operations on C strings. Programmers can also define their own string classes with appropriate member functions to manipulate strings. ANSI standard C++ introduces a new class called string which is an improvised version of C strings in several ways. In many cases, the strings object may be treated like any other built-in data type. The string is treated as another container class for C++.

1.The C Style String:

The C style string belongs to C language and continues to support in C++ also strings in C are the one-dimensional array of characters which gets terminated by \0 (null character).

This is how the strings in C are declared:
char ch[6] = {'H', 'e', 'l', 'l', 'o', '

char ch[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

'};

2. String Class in C++:

The string class is huge and includes many constructors, member functions, and operators.

Programmers may use the constructors, operators and member functions to achieve the following:

- Creating string objects
- Reading string objects from keyboard
- Displaying string objects to the screen
- Finding a substring from a string
- Modifying string
- Adding objects of string
- Comparing strings
- Accessing characters of a string
- Obtaining the size or length of a string, etc...

3.Manipulate Null-terminated strings:

C++ supports a wide range of functions that manipulate null-terminated strings. These are:

- strcpy(str1, str2): Copies string str2 into string str1.
- strcat(str1, str2): Concatenates string str2 onto the end of string str1.
- strlen(str1): Returns the length of string str1.
- strcmp(str1, str2): Returns 0 if str1 and str2 are the same; less than 0 if str1<str2; greater than 0 if str1>str2.
- strchr(str1, ch): Returns a pointer to the first occurrence of character ch in string str1.
- strstr(str1, str2): Returns a pointer to the first occurrence of string str2 in string str1.

4.Important functions supported by String Class:

- append(): This function appends a part of a string to another string
- assign():This function assigns a partial string
- at(): This function obtains the character stored at a specified location
- begin(): This function returns a reference to the start of the string
- capacity(): This function gives the total element that can be stored
- compare(): This function compares a string against the invoking string
- empty(): This function returns true if the string is empty

- end(): This function returns a reference to the end of the string
- erase(): This function removes character as specified
- find(): This function searches for the occurrence of a specified substring
- length(): It gives the size of a string or the number of elements of a string
- swap(): This function swaps the given string with the invoking one

5.Important Constructors obtained by String Class:

- String(): This constructor is used for creating an empty string
- String(const char *str): This constructor is used for creating string objects from a null-terminated string
- String(const string *str): This constructor is used for creating a string object from another string object

6.Operators used for String Objects:

1. =: assignment
2. +: concatenation
3. ==: Equality
4. !=: Inequality
5. <: Less than
6. <=: Less than or equal
7. >: Greater than
8. >=: Greater than or equal
9. []: Subscription
10. <<: Output
11. >>: Input

PART-C

1 .Explain the phases in Software Development Life Cycle (SDLC)?

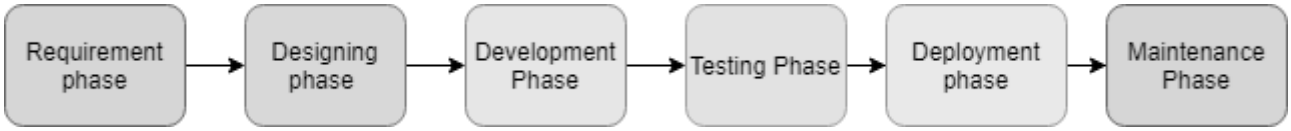
Software Development Life Cycle (SDLC):

SDLC is a process that creates a structure of development of software. There are different phases within SDLC, and each phase has its various activities. It makes the development team able to design, create, and deliver a high-quality product.

SDLC describes various phases of software development and the order of execution of phases. Each phase requires deliverable from the previous phase in a life cycle of software development. Requirements are translated into design, design into development and development into testing; after testing, it is given to the client.

Let's see all the phases in detail:

Different phases of the software development cycle



- **Requirement Phase**
- **Design Phase**
- **Build /Development Phase**
- **Testing Phase**
- **Deployment/ Deliver Phase**
- **Maintenance**

1. Requirement Phase

This is the most crucial phase of the software development life cycle for the developing team as well as for the project manager. During this phase, the client states requirements, specifications, expectations, and any other special requirement related to the product or software. All these are gathered by the business manager or project manager or analyst of the service providing company.

The requirement includes how the product will be used and who will use the product to determine the load of operations. All information gathered from this phase is critical to developing the product as per the customer requirements.

2. Design Phase

The design phase includes a detailed analysis of new software according to the requirement phase. This is the high priority phase in the development life cycle of a system because the logical designing of the system is converted into physical designing. The output of the requirement phase is a collection of things that are required, and the design phase gives the way to accomplish these requirements. The decision of all required essential tools such

as **programming language** like Java, .NET, PHP, a **database** like Oracle, MySQL, a combination of hardware and software to provide a platform on which software can run without any problem is taken in this phase.

There are several techniques and tools, such as data flow diagrams, flowcharts, decision tables, and decision trees, Data dictionary, and the structured dictionary are used for describing the system design.

3. Build /Development Phase

After the successful completion of the requirement and design phase, the next step is to implement the design into the development of a software system. In this phase, work is divided into small units, and coding starts by the team of developers according to the design discussed in the previous phase and according to the requirements of the client discussed in requirement phase to produce the desired result.

Front-end developers develop easy and attractive GUI and necessary interfaces to interact with back-end operations and back-end developers do back-end coding according to the required operations. All is done according to the procedure and guidelines demonstrated by the project manager.

Since this is the coding phase, it takes the longest time and more focused approach for the developer in the software development life cycle.

4. Testing Phase

Testing is the last step of completing a software system. In this phase, after getting the developed GUI and back-end combination, it is tested against the requirements stated in the requirement phase. Testing determines whether the software is actually giving the result as per the requirements addressed in the requirement phase or not. The Development team makes a test plan to start the test. This test plan includes all types of essential testing such as integration testing, unit testing, acceptance testing, and system testing. Non-functional testing is also done in this phase.

If there are any defects in the software or it is not working as per expectations, then the testing team gives information to the development team in detail about the issue. If it is a valid defect or worth to sort out, it will be fixed, and the development team replaces it with the new one, and it also needs to be verified.

5. Deployment/ Deliver Phase

When software testing is completed with a satisfying result, and there are no remaining issues in the working of the software, it is delivered to the customer for their use.

As soon as customers receive the product, they are recommended first to do the beta testing. In beta testing, customer can require any changes which are not present in the software but mentioned in the requirement document or any other GUI changes to make it more user-friendly. Besides this, if any type of defect is encountered while a customer using the software; it will be informed to the development team of that particular software to sort out

the problem. If it is a severe issue, then the development team solves it in a short time; otherwise, if it is less severe, then it will wait for the next version.

After the solution of all types of bugs and changes, the software finally deployed to the end-user.

6. Maintenance

The maintenance phase is the last and long-lasting phase of SDLC because it is the process which continues until the software's life cycle comes to an end. When a customer starts using software, then actual problems start to occur, and at that time there's a need to solve these problems. This phase also includes making changes in hardware and software to maintain its operational effectiveness like to improve its performance, enhance security features and according to customer's requirements with upcoming time. This process to take care of product time to time is called maintenance.

"So, all these are six phases of software development life cycle (SDLC) under which the process of development of software takes place. All are compulsory phases without any one of the development cannot be possible because development continues for the lifetime of software with maintenance phase".

Software Development Life Cycle (SDLC) Models

The software development models are those several process or approaches which are being selected for the development of project based on the project's objectives. To accomplish various purposes, many development life cycle models. And these models identify the multiple phases of the process. Picking up the correct model for developing the software application is very important because it will explain the what, where, and when of our planned testing.

Here, are various software development models or methodologies:

- **Waterfall model**
- **Spiral model**
- **Verification and validation model**
- **Prototype model**
- **Hybrid model**

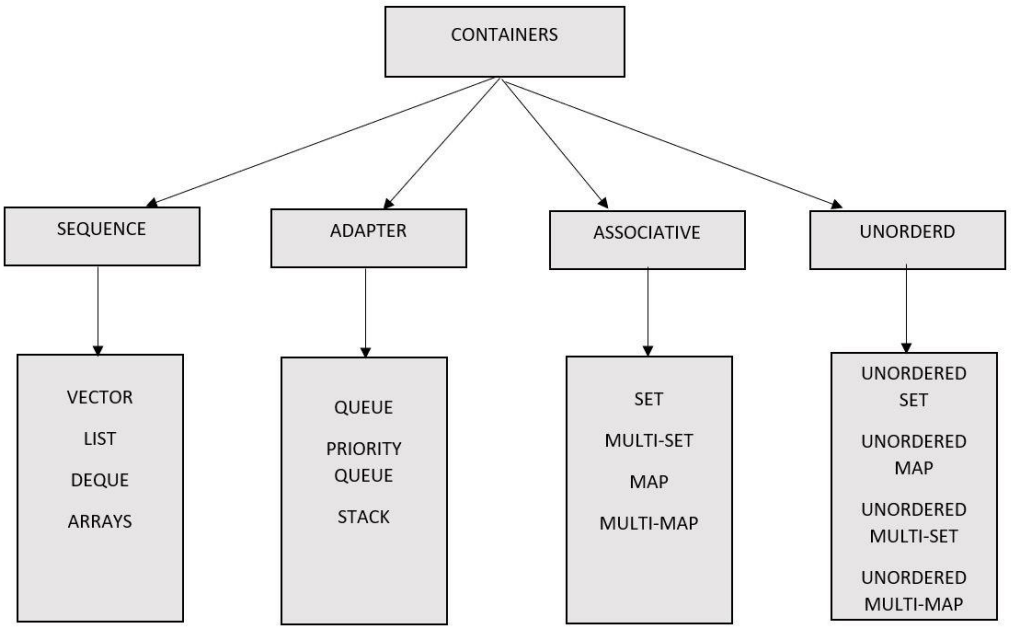
2.Explain the component of STL?

CONTAINERS

Containers can be described as the objects that hold the data of the same type. Containers are used to implement different data structures for example arrays, list, trees, etc.

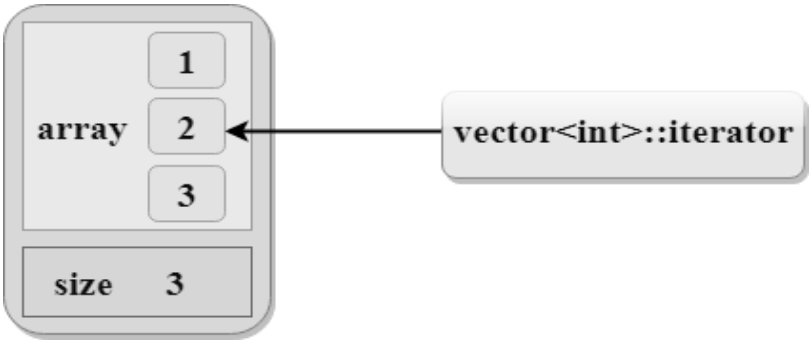
Classification of containers :

- Sequence containers
- Associative containers
- Derived containers



ITERATOR

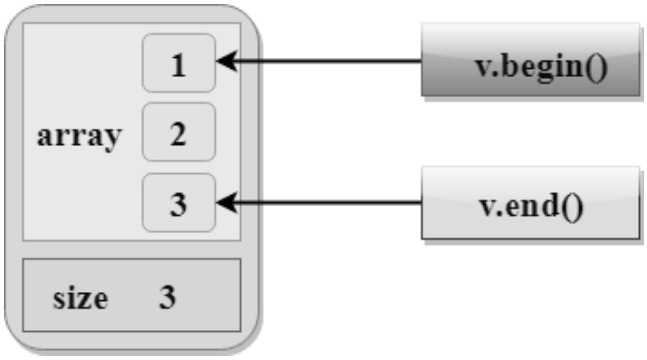
- Iterators are pointer-like entities used to access the individual elements in a container.
- Iterators are moved sequentially from one element to another element. This process is known as iterating through a container.



- Iterator contains mainly two functions:

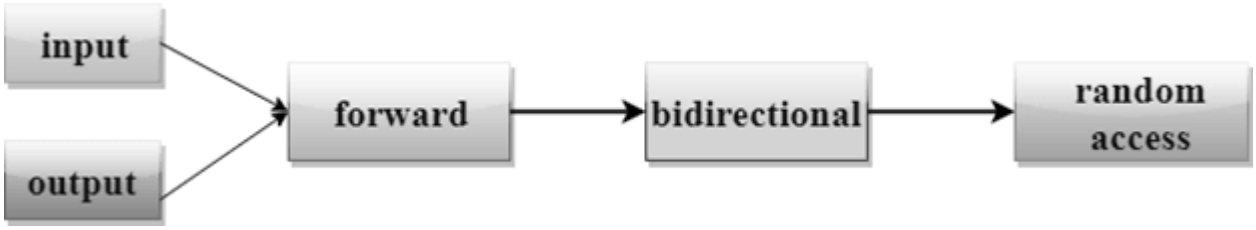
begin(): The member function begin() returns an iterator to the first element of the vector.

end(): The member function end() returns an iterator to the past-the-last element of a container.



Iterator Categories

Iterators are mainly divided into five categories:



1. **Input iterator:**
   - An Input iterator is an iterator that allows the program to read the values from the container.
   - Dereferencing the input iterator allows us to read a value from the container, but it does not alter the value.
   - An Input iterator is a one way iterator.
   - An Input iterator can be incremented, but it cannot be decremented.

2. **Output iterator:**
    - An output iterator is similar to the input iterator, except that it allows the program to modify a value of the container, but it does not allow to read it.
    - It is a one-way iterator.
    - It is a write only iterator.
3. **Forward iterator:**
    - Forward iterator uses the ++ operator to navigate through the container.
    - Forward iterator goes through each element of a container and one element at a time.
4. **Bidirectional iterator:**
    - A Bidirectional iterator is similar to the forward iterator, except that it also moves in the backward direction.
    - It is a two way iterator.
    - It can be incremented as well as decremented.
5. **Random Access Iterator:**
    - Random access iterator can be used to access the random element of a container.
    - Random access iterator has all the features of a bidirectional iterator, and it also has one more additional feature, i.e., pointer addition. By using the pointer addition operation, access the random element of a container.

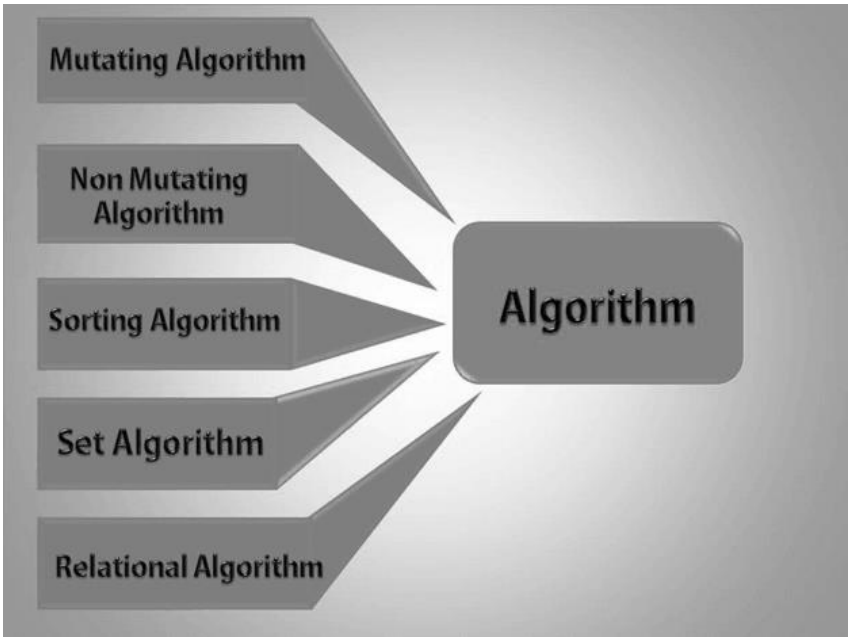
Algorithms

Algorithms are the functions used across a variety of containers for processing its contents.

Points to Remember:

- Algorithms provide approx 60 algorithm functions to perform the complex operations.
- Standard algorithms allow us to work with two different types of the container at the same time.
- Algorithms are not the member functions of a container, but they are the standalone template functions.
- Algorithms save a lot of time and effort.
- In want to access the STL algorithms, must include the <algorithm> header file in our program.

STL algorithms can be categorized as:



- **Nonmutating algorithms**: Nonmutating algorithms are the algorithms that do not alter any value of a container object nor do they change the order of the elements in which they appear. These algorithms can be used for all the container objects, and they make use of the forward iterators.
- **Mutating algorithms**: Mutating algorithms are the algorithms that can be used to alter the value of a container. They can also be used to change the order of the elements in which they appear.
- **Sorting algorithms**: Sorting algorithms are the modifying algorithms used to sort the elements in a container.
- **Set algorithms**: Set algorithms are also known as sorted range algorithm. This algorithm is used to perform some function on a container that greatly improves the efficiency of a program.
- **Relational algorithms**: Relational algorithms are the algorithms used to work on the numerical data. They are mainly designed to perform the mathematical operations to all the elements in a container.